IBM

IBMパーソナルシステム/55™

BASIC インタープリター ユーザーズ・ガイド （上巻）解説編

N:SH18-2120-2

IBM

IBMパーソナルシステム/55™

BASIC インタープリター
ユーザーズ・ガイド
（上巻）解説編

N:SH18-2120-2

第 3 版　1989 年 5 月

このマニュアルは、製品の改良その他により適宜改訂されます。

© Copyright International Business Machines Corporation 1986,1989

はじめに

BASIC インタープリター　ユーザーズ・ガイドは，IBM マルチステーション 5550 で働く BASIC インタープリター バージョン K3.2 と K3.3 の使用方法や文法を詳細に説明した，参照用のマニュアルです。

このユーザーズ・ガイドは次の各編より成る 2 分冊で構成されています。

- 上巻 → 解説編
- 下巻 → 文法編

本書は**解説編**です。この解説編は，別冊の文法編と併せてお使いください。本書で「文法編」と記述しているときは，BASIC インタープリター ユーザーズ・ガイドの下巻「文法編」を指します。

この**解説編**では，主として BASIC でできること，BASIC を使用する上での基本的な規則を説明しています。また，付録では，BASIC の持つ機能を活用するときに有効な情報を載せています。

文法編では，BASIC のコマンド，ステートメント，関数，変数をアルファベット順に説明しています。付録にはメッセージ集があります。

バージョンK3.2とK3.3の機能

BASIC バージョン K3.2 と K3.3 で追加または拡張された機能は次のとおりです。なお，バージョン K2.0 と K2.1 で追加された機能については，文法編「BASIC の文法」の最後に「バージョン・アップの追加機能一覧」がありますので，これを参照してください。

バージョンK3.2の追加/拡張機能の概要

1 共用ファイルに対するアクセスの制御

- LOCK，UNLOCK ステートメント
- OPEN ステートメント

複数の処理プログラムが入出力を行うようなファイルをアクセスする場合に，他のプログラムによるアクセスをどのように制御するかまたは，他のプログラムが使用しているファイルに対してどのようにオープンするかを設定できます。

2 キー処理の拡張

- ON KEY (n)，KEY (n) ステートメント
- KEY ステートメント

PF11 と PF12 にも PF1 ～ PF10 と同じようにキー機能を割り当てることができます。また独自のキー割り込みを定義するための，キー操作に対応する番号が 5 つ増えています。KEY (n) は，KEY (0) と指定すると，定義されたすべての割り込みキーを定義できます。

3 その他の機能拡張

- EXTERR (n) 関数

DOS の拡張エラー情報を得ることができます。

- ERDEV 関数

装置ヘッダーの属性ビットが追加されています。

- レコード番号の拡張

レコード番号が 1 ～ 4,294,967,295 (2^{32}-1) まで使用できます。

4 JIS83 年版文字セットの使用

DOS で JIS83 年版文字セットが指定されているときは，BASIC でも JIS83 年版文字セットが使用できます。

バージョン K3.3 について

バージョン K3.3 では，グラフィックス支援カードを使用すると，16 色カラー・グラフィックス・モードで，より高速なグラフィックス処理が可能です。各機能はバージョン K3.2 と同じです。バージョン K3.3 で高速グラフィックス処理を行うためには，必ずグラフィックス支援カードが要ります。

本書の構成

解説編

BASIC ユーザーズ・ガイド解説編は次の各章から構成されています。

第 1 章 BASIC の始動と操作の方法

BASIC の始動（終了），キーの操作，プログラムの編集などについて説明しています。BASIC インタープリターは会話形式でプログラムを作成していきます。BASIC の操作に不慣れな方や，他言語の知識をお持ちの方でインタープリターは初めての方は特に確認のために参照してください。

第 2 章 プログラミングの基本規則

BASIC で使用できる文字，定数や変数における文字や数値の表現の規則，演算子，いわば BASIC 規則集です。ただし，規則だけでなく使い方についても例と共に説明しています。

第 3 章 BASIC の入出力指定

この章では，BASIC でファイルや画面に対して入出力を行うときに，知っておく必要がある事柄を説明しています。画面の使い方では画面モードの機能や色の使い方を説明しています。

付録 A.ディスク入出力

BASIC では「順次ファイル」と「ランダム・ファイル」を作成することができます。この付録ではこの 2 つのファイルの作成やデータの検索を行うプログラムを記述するためには，どのような手順を踏むべきかについて例を挙げて説明しています。その他，プログラム・ファイルを扱うコマンドについても紹介しています。

付録 B.機械語サブルーチン

アセンブラーで記述したサブルーチンを，BASIC から呼び出すこともできます。その場合，記憶域の内容について配慮が必要となります。ここでは，その方法について説明しています。

付録 C.通信

BASIC で，非同期通信を行う場合，ディスク・ファイルとほとんど同じようにデータの入出力を行うことができます。ただし，データをやりとりする場合のタイミングに注意を払う必要があります。ここでは，通信固有のことがらを中心に，BASIC で通信を行うためのプログラムの記述方法について説明しています。

付録 D.コード体系

BASIC で使用する文字には，すべて一連のコードが振られています。この付録では，これらのコード体系の相互関係を図で示して説明しています。BASIC ではキーの操作を操作に対する番号で得ることもできます。また，出力装置に制御文字を送ることもできます。これらの方法は文法編と併せて参照してください。

付録 E.技術情報

記憶域マップ，変数の値の格納方法，BASIC ファイル制御ブロックなどの技術情報を提供しています。

付録 F.グラフィックス

ここでは，「円グラフ」「棒グラフ」「絵グラフ」のプログラム例をもとに，BASIC のグラフィック機能を紹介しています。BASIC のグラフィック・ステートメントの使用例として参照してください。

付録 G.用語集

この用語集は BASIC でプログラミングする場合に一般的に良く用いられる用語や本書で使用している特別の用語の意味を「あいうえお」順で説明しています。

文法編

別冊の BASIC インタープリター・ユーザーズ・ガイド文法編は次のように構成されています。

BASIC の文法

BASIC のコマンド，ステートメント，関数，変数をアルファベット順に詳しく説明しています。使用するステートメントなどの名前が不明でどこを参照するべきか分からない場合は，目的別索引（ユーザーズ・ガイドの最後に記

載）または文法編の「はじめに」にある「機能別ステートメント一覧」を参照してください。

付録 A.メッセージ集

BASIC 使用中に画面に出力されるメッセージの意味とその対応について説明しています。

付録 B.参考資料

プログラミングする場合に役立つことがらを一覧表にして記載しています。この内容のほとんどの項目は解説編で説明している内容を簡略した表に直したものです。内容について不明な点は解説編または文法編の関連項目を参照してください。

本書の参照方法

BASIC インタープリターで初めてプログラミングされる方は,

解説編の第 1 章と第 2 章, 第 3 章は BASIC の操作と規則を知るための説明を記述しています。すべてを覚える必要はありませんが, 簡単に一読して情報の記述場所を確認してください。その後, 文法編を参照してください。プログラミングの目的から使用するステートメントなどを見つけるには, ユーザーズ・ガイドの最後にある目的別索引を参照してください。解説編の付録は必要に応じて参照してください。

BASIC インタープリターによるプログラミングに慣れている方は,

文法編の「はじめに」にある BASIC の概要または目的別索引は使用するステートメントを確認するために便利です。プログラミングのときは「BASIC の文法」を参照してください。解説編の付録は, 各機能を活用するために随時参照してください。

関連マニュアル

システム操作ガイド, N:GA18-2325

日本語 DOS ユーザーズ・ガイド　入門編, N:SC18-2135
日本語 DOS ユーザーズ・ガイド　参照編, N:SC18-2136
日本語 DOS の内部構造とプログラム・インターフェース, N:SC18-2128
日本語文節変換プログラム　ユーザーズガイド, N:SC18-2119
ハード・ディスクをお使いいただくために, N:SC18-2004

IBM マルチステーション 5550 BASIC の操作法, N:SR18-3785
IBM マルチステーション 5550 BASIC のプログラミング, N:SR18-3780

漢字コード一覧表, N:GC18-2040

BASICインタープリターとは

BASICについて

BASICは広く普及しているプログラム言語ですが，BASICの構文はハードウェア（コンピューター）に依存する場合が多く，したがって使用するハードウェアによってさまざまな構文を有する言語でもあります。本システムのBASICは次の機能を提供しています。

- ディスク・ファイルの入出力
- キーボード入力や画面/プリンターへの出力
- 図形処理
- 非同期通信
- 機械語サブルーチンの呼び出し
- 音楽演奏
- 刻時（日付と時間の入出力）

本システムのBASICには次の特徴があります。

- ディスク・ファイル，その他の入出力装置や通信ファイルに対して，すべて同じような形式でデータ入出力の処理を行うことができます（BASICによるデータのやりとりの他に対象となる装置に特有な処理は必要となります）。

- BASICでは24ドットのシステムで1024 × 768，16ドットのシステムで720 × 512の高解像度のグラフィック・モード画面を使用できます。また色は64色の中からで同時に最高16色までを表示できます。線や円を描く基本的な図形処理機能のほか，閉じた図形を模様で埋めたり，表示範囲や表示物の大きさを自由に変更する機能もあります。

- BASICで通信用のプログラムを作成し，別のコンピューターとデータの転送やファイルの転送を行うことができます。

- 曲の施律を演奏できます。プログラム実行中に背景音楽（BGM）を演奏させることができます。音は周波数や発生時間でも指定できるため，曲以外のさまざまな音を表現することもできます。

- 本システムには刻時装置が装備されているため，日時の設定や現在の時刻を知ることができます。タイマーとして使用することもできます。

インタープリターとは BASICで記述されたプログラムは機械語の命令に翻訳するために，通常一括して翻訳するか，1行ごとに翻訳して実行するかの，どちらかの方法をとります。前者の処理を行う翻訳プログラム（この場合はBASIC）はコンパイラー，後者の翻訳プログラムはインタープリターと呼ばれています。

インタープリターは1行ごとに翻訳し，ステートメントを分析して，エラーがあるか否かを調べるとともにステートメントで記述された機能を実行します。

このBASICインタープリターはプログラム編集機能があり，会話形式でプログラムを作成，修正できます。またインタープリターであるため，プログラム実行後の変数の内容を調べたり，BASICの機能を試してみることが，比較的容易にできます。

ハードウェア構成

BASIC インタープリターを使用するためには，次のいずれかのハードウェアが必要です。

- □ システム・ユニット‥‥ 1ディスケット・ドライブ以上，256KB 以上
- □ キーボード‥‥‥‥‥‥ Ⅰ～Ⅳのいずれか
- □ ディスプレイ‥‥‥‥‥ 12 型単色，14 型カラー，15 型単色，16 型カラー
- □ プリンター

プリンターは印刷出力を必要とする場合にのみ必要となります。

BASIC を使用するためには最低 256K バイトの記憶域が必要ですが，IBM 5550 は最少のシステムでも標準装備で 256K バイトあるため，十分使用できます。

解説編の目次

第1章 BASICの始動と操作の方法

BASICの始動 …… 1-1
　日付と時刻の入力 …… 1-1
　DOSプロンプト …… 1-1
　BASICコマンド …… 1-1
　DOSに戻る …… 1-1
　システム・リセット …… 1-2
　ハード・ディスク …… 1-2
キー操作 …… 1-3
文字を入力するキー …… 1-4
　英数字の入力 …… 1-4
　カタカナの入力 …… 1-5
　ひらがなの入力 …… 1-5
　漢字の入力 …… 1-5
　数値の入力 …… 1-10
プログラムを編集するキー …… 1-11
　プログラムを入力する …… 1-11
　カーソル移動キー …… 1-11
　編集キー …… 1-12
　実行を制御するキー …… 1-13
　予約語を表示するキー …… 1-13
　印刷の指定を行うキー …… 1-14
BASICのプログラム …… 1-15
　行番号 …… 1-16
　ステートメント …… 1-16
　注釈 …… 1-17
プログラムの作成と実行 …… 1-19
　プログラムの作成 …… 1-19
　プログラムの実行 …… 1-19

プログラムの保管 …… 1-19
プログラムの読み込み …… 1-19
リストの表示 …… 1-20
プログラム実行のモード …… 1-21
直接モード …… 1-21
間接モード …… 1-21
プログラムの修正 …… 1-22
文字の変更 …… 1-23
文字の削除 …… 1-23
文字の挿入 …… 1-24
行の消去 …… 1-24
行の追加 …… 1-24
行の削除 …… 1-26
プログラムの削除 …… 1-26
構文エラー …… 1-27
変数の内容を調べる …… 1-28
BASIC 始動時に指定できる項目 …… 1-29

第2章 プログラミングの基本規則

BASIC で使用できる文字 …… 2-1
特殊文字 …… 2-2
予約語 …… 2-3
BASIC で使用できるデータ …… 2-6
定数 …… 2-7
ストリング定数 …… 2-7
数値定数 …… 2-7
整数 …… 2-7
固定小数点数 …… 2-7
浮動小数点数 …… 2-8
16 進数 …… 2-8
8 進数 …… 2-9
JIS16 進数 …… 2-9
JIS 区点定数 …… 2-9

変数 …… 2-10
変数名 …… 2-11
変数型 …… 2-11
数値精度 …… 2-13
数値精度の変換 …… 2-15
配列 …… 2-18
数式の演算子 …… 2-21
算術演算子 …… 2-22
整数除算 …… 2-22
剰余の計算 …… 2-23
関数演算子 …… 2-24
数値比較 …… 2-24
ストリング比較 …… 2-25
論理演算子 …… 2-26
論理演算子の働き …… 2-28
実行順序 …… 2-31
ストリングの演算子 …… 2-33
連結演算子 …… 2-34

第3章 BASICの入出力指定

ファイル …… 3-1
ファイル番号 …… 3-2
ファイル指定とパス …… 3-2
ファイル指定 …… 3-2
パス名 …… 3-3
装置名 …… 3-4
ファイル名 …… 3-5
ディレクトリー …… 3-6
ディレクトリーの構造 …… 3-6
ディレクトリーのタイプ …… 3-7
現行ディレクトリー …… 3-7
画面の使用 …… 3-8

画面モード …… 3-9
単色システム …… 3-10
カラー・システム …… 3-11
文字モード …… 3-12
グラフィック・モード …… 3-14
座標指定 …… 3-14
ウインドーとビューポート …… 3-16
色 …… 3-21
色の使用と画面モード …… 3-26
単色文字モード …… 3-26
カラー文字モード …… 3-27
単色グラフィック・モード …… 3-28
4色カラー・グラフィック・モード …… 3-29
16色カラー・グラフィック・モード …… 3-30
画面表示 …… 3-33

付録A ディスク入出力

プログラム・ファイルの管理 …… A-1
プログラムの保管 …… A-1
プログラムの読み込み …… A-2
プログラムの実行 …… A-2
重ね合わせ …… A-3
ファイルの削除 …… A-4
ファイル名の変更 …… A-4
順次ファイル …… A-5
順次ファイルの書き込み …… A-7
順次ファイルの読み取り …… A-9
順次ファイルのプログラム例 …… A-10
プログラムの目的 …… A-10
出力例 …… A-10
プログラム・リスト 1 …… A-11
プログラム・リスト 2 …… A-12
ランダム・ファイル …… A-15
ランダム・ファイルの書き込み …… A-18
ランダム・ファイルの読み取り …… A-20

ランダム・ファイルのプログラム例 …… A-22
プログラムの目的 …… A-22
プログラム・リスト …… A-23

付録B 機械語サブルーチン

機械語サブルーチンの処理概略 …… B-1
サブルーチンを記憶域に置く …… B-2
なぜ整数配列なのか …… B-2
サブルーチン・ファイル …… B-2
小さいサブルーチン …… B-3
READで …… B-4
POKEで …… B-6
サブルーチン・ファイル1 …… B-8
値の受け渡し1 …… B-10
値の受け渡し2 …… B-15
サブルーチンの呼び出し …… B-18
CALLステートメント …… B-20
USR関数 …… B-22

付録C 通信

通信ファイルの入出力 …… C-1
送信停止と送信開始 …… C-2
入出力関数 …… C-2
入力ステートメント …… C-3
通信端末 …… C-4
ファイルの送受信 …… C-6
プログラミングの注意 …… C-8
半2重と全2重 …… C-8
交換回線と専用回線 …… C-8
かな/漢字の通信 …… C-8
雑音対策 …… C-9
制御信号の操作 …… C-10
OPEN時の出力信号 …… C-10

入力制御信号 …… C-10
モデム装置制御信号 …… C-11
出力制御信号 …… C-13
通信エラー …… C-15
ファイルのオープン時 …… C-15
データの読み取り時 …… C-15
データの書き込み時 …… C-15

付録D コード体系

コード体系 …… D-1
連続コード（10 進数） …… D-1
内部コード（16 進数） …… D-1
JIS 区点コード（10 進数） …… D-1
JIS16 進コード（16 進数） …… D-1
1 バイト・コード文字セット …… D-3
制御文字と連続コード …… D-8
コード間の変換 …… D-10
2 バイト・コード文字セット …… D-11
連続コードと内部コードの関係 …… D-12
JIS 区点コードと内部コードの関係 …… D-13
JIS16 進コードと内部コードの関係 …… D-15
コード間の変換 …… D-16
拡張コード …… D-19

付録E 技術情報

記憶域マップ …… E-1
変数の記憶方法 …… E-3
ストリング変数の記憶 …… E-3
数値変数の記憶 …… E-4
BASIC のファイル制御ブロック …… E-5
キーボード・バッファー …… E-7
印刷制御文字の指定 …… E-8
プログラム例 1 …… E-10
プログラム例 2 …… E-11

付録F グラフィックス

BASIC のグラフィックス …… F-1
円グラフ …… F-2
　プログラムの目的 …… F-2
　出力例 …… F-2
　プログラム・リスト …… F-3
プログラムの処理内容 …… F-7
　システムを調べる …… F-7
　変数の初期設定 …… F-8
　データの入力 …… F-8
　百分率を求める …… F-9
　角度を求める …… F-9
　円グラフを描く …… F-9
　円グラフを塗る …… F-9
　項目名を表示する …… F-9
　項目名と百分率を表示 …… -10
　プログラムの終り …… F-10
棒グラフ …… F-11
　プログラムの目的 …… F-11
　出力例 …… F-11
　プログラム・リスト …… F-12
プログラムの処理内容 …… F-15
　軸を描く …… F-15
　データの入力 …… F-16
　棒を描く …… F-16
絵グラフ …… F-17
　プログラムの目的 …… F-17
　出力例 …… F-17
　プログラム・リスト …… F-18
プログラムの処理内容 …… F-21
　車を表示するサブルーチン …… F-22
　値に従って車を表示 …… F-22

付録G 用語集

文法編の目次

BASICの文法

ABS …… 絶対値 …… 1
ASC …… 連続コード …… 2
ATN …… 逆正接 …… 3
AUTO …… 行番号の自動作成 …… 4
BEEP …… スピーカーを鳴らす …… 6
BLOAD …… 記憶域イメージ・ファイルの読み込み …… 7
BSAVE …… 記憶域イメージ・ファイルの保管 …… 9
CALL …… 機械語サブルーチンの呼び出し …… 10
CDBL …… 倍精度に変換 …… 11
CDBL$ …… 1バイト・コード英数字を2バイト・コードに …… 12
CHAIN …… 他プログラムに制御を移す …… 13
CHDIR …… 現行ディレクトリーの変更 …… 16
CHR$ …… 連続コードを文字に変換 …… 18
CINT …… 整数に変換 …… 21
CIRCLE …… 円を描く …… 22
CLEAR …… データの消去 …… 26
CLOSE …… ファイルのクローズ …… 28
CLS …… 画面の消去 …… 30
COLOR …… 文字の色と属性，図形や背景の色指定 …… 32
COM (n) …… 通信割り込み処理の制御 …… 42
COMMON …… 連鎖されたプログラムに制御を渡す …… 43
CONT …… 実行の再開 …… 44
COS …… 余弦 …… 46
CSNG …… 単精度に変換 …… 47
CSNG$ …… 2バイト・コード英数字を1バイト・コードに …… 48
CSRLIN …… 現在のカーソルの行位置 …… 49
CVI, CVS, CVD …… ストリング型から数値型に変換 …… 50
DATA …… 数値定数やストリング定数の記憶 …… 52

DATE$ ······ 日付の読み取りと設定 ······ 53
DEF FN ······ ユーザー関数の定義 ······ 55
DEF SEG ······ セグメントの定義 ······ 57
DEF 型 ······ 変数の型を宣言する ······ 58
DEF USR ······ 機械語サブルーチンの開始アドレスの指定 ······ 60
DELETE ······ 行の削除 ······ 61
DIM ······ 配列の宣言 ······ 62
DRAW ······ 線を描く ······ 64
EDIT ······ 行の表示と編集 ······ 69
END ······ 実行の終了 ······ 70
ENVIRON ······ 環境にパラメーターを設定する ······ 71
ENVIRON$ ······ 環境から値を読み取る ······ 73
EOF ······ ファイルの終了状態を返す ······ 75
ERASE ······ 配列の消去 ······ 76
ERDEV, ERDEV$ ······ 装置のエラー情報 ······ 77
ERR, ERL ······ エラー・コードと行番号を得る ······ 79
ERROR ······ エラー発生の模擬とエラー・コード定義 ······ 81
EXP ······ 指数関数 ······ 83
EXTERR (n) ······ 拡張エラー情報を得る ······ 84
FIELD ······ レコードの項目定義 ······ 86
FILES ······ ファイル名の表示 ······ 89
FIX ······ 整数変換 ······ 91
FOR, NEXT ······ 処理の繰り返し ······ 92
FRE ······ 記憶域の未使用空間のバイト数を得る ······ 96
GET (ファイル) ······ レコードの読み込み ······ 97
GET (グラフィック) ······ 複数の点の読み取り ······ 99
GOSUB ······ サブルーチンに行き, 戻る ······ 101
GOTO ······ 分岐 ······ 103
HEX$ ······ 10 進数を 16 進数に変換 ······ 104
IF ······ 判断 ······ 105
INKEY$ ······ キーボードから文字を読み取る ······ 108
INP ······ 入力ポートの 1 バイトを返す ······ 110
INPUT ······ キーボード入力 ······ 111
INPUT# ······ 順次ファイルから変数に読み取る ······ 114

INPUT$ …………………… ファイルから文字を読み取る …………………… 116
INPUT¥ …………………… ファイルから文字を読み取る …………………… 118
INSTR …………………… ストリング内の探索 …………………… 119
INT …………………… 整数変換 …………………… 121
IOCTL …………………… 装置駆動ルーチンへ制御データを渡す …………………… 122
IOCTL$ …………………… 装置駆動制御データ実行確認 …………………… 123
JIS$ …………………… JIS16進コードを返す …………………… 124
KEY …………………… PFキー表示，割り込み処理キーの設定 …………………… 125
KEY (n) …………………… キーの割り込み処置の制御 …………………… 130
KILL …………………… ファイルの削除 …………………… 132
KLEN …………………… 文字数を返す …………………… 133
KPOS …………………… ストリング内の文字位置を返す …………………… 134
KTN$ …………………… JIS区点コードを返す …………………… 135
LEFT$ …………………… 左から指定バイト数の文字を返す …………………… 136
LEN …………………… 文字のバイト数を返す …………………… 138
LET …………………… 式の値を変数に代入する …………………… 139
LINE …………………… 線，矩形を描く …………………… 140
LINE INPUT …………………… キーボードから1行読み込む …………………… 144
LINE INPUT# …………………… 順次ファイルから1行読み込む …………………… 145
LIST …………………… プログラムの表示 …………………… 147
LLIST …………………… プリンター出力 …………………… 149
LOAD …………………… プログラムの読み込み …………………… 150
LOC …………………… ファイル内の現在位置を得る …………………… 152
LOCATE …………………… カーソル位置とカーソル表示形態の指定 …………………… 153
LOCK …………………… レコードの保護指定 …………………… 156
LOF …………………… ファイルの長さを返す …………………… 157
LOG …………………… 自然対数 …………………… 158
LPOS …………………… プリント・ヘッドの現在位置を得る …………………… 159
LPRINT …………………… プリンター印刷 …………………… 160
LSET, RSET …………………… ストリングの左寄せ，右寄せ …………………… 162
MERGE …………………… プログラムのマージ …………………… 163
MID$ …………………… ストリングの置き換え …………………… 164
MID$ …………………… ストリングの一部を返す …………………… 166
MKDIR …………………… ディレクトリーの作成 …………………… 168

MKI$, MKS$, MKD$ ………… 数値からストリングに変換 ………… 169
NAME………… ファイル名の変更 ………… 170
NEW………… 記憶域内のプログラムや変数の消去 ………… 171
NEXT………… ループを閉じる ………… 172
OCT$ ………… 10進数を8進数に変換………… 173
ON COM (n) ………… 通信割り込み処理ルーチンの指定 ………… 174
ON ERROR………… エラー発生時処理ルーチンの指定 ………… 176
ON GOSUB (GOTO) ………… サブルーチンまたは行への分岐 ………… 177
ON KEY (n) ………… キー入力処理ルーチンの指定 ………… 178
ON PLAY ………… 音が一定以下の時の処理ルーチンの指定 ………… 180
ON TIMER (n) ………… 時間経過時処理ルーチンの指定 ………… 183
OPEN ………… ファイルのオープン ………… 185
OPEN"COM… ………… 通信ファイルのオープン ………… 192
OPTION BASE ………… 配列添字の最小値指定 ………… 198
OUT………… 出力ポートへ1バイト送る ………… 199
PAINT ………… 色や模様を塗る ………… 200
PALETTE ………… パレットに色を設定する ………… 206
PEEK ………… 記憶域から1バイト読み取る ………… 208
PLAY ………… 音楽演奏 ………… 209
PLAY (n) ………… 音楽バッファーの音の数を返す ………… 214
PMAP………… ウインドーとビューポート間の座標変換 ………… 215
POINT ………… 点の色を返す ………… 216
POKE ………… 記憶域に1バイト書き込む ………… 218
POS ………… カーソルの桁位置を返す ………… 219
PRESET ………… 点を描く ………… 220
PRINT ………… 画面にデータを表示する ………… 221
PRINT USING………… 指定された書式で画面にデータを表示 ………… 224
PRINT# ………… データをファイルへ書き込む ………… 229
PSET ………… 点を描く ………… 232
PUT (ファイル) ………… レコードの書き込み ………… 234
PUT (グラフィック) ………… 図形を画面に設定する ………… 235
RANDOMIZE………… 乱数シードの再設定 ………… 241
READ………… DATA ステートメントから値を読み取る ………… 243
REM………… 注釈行の指定 ………… 245

RENUM	行番号の変更	246
RESET	ファイルのクローズとバッファーの消去	248
RESTORE	DATA ステートメントの行指定	249
RESUME	再実行	250
RETURN	サブルーチンから戻る	252
RIGHT$	ストリングの右から任意バイトの文字を返す	253
RMDIR	ディレクトリーの消去	255
RND	乱数発生	257
RSET	ストリングの右よせ	259
RUN	プログラムの実行	260
SAVE	プログラムの保管	262
SCREEN	文字の属性を返す	264
SCREEN	画面モードの設定	267
SGN	符号を返す	270
SHELL	DOS コマンドの実行	271
SIN	正弦	273
SOUND	音の発生	274
SPACE$	ストリングのスペース指定	277
SPC	スペースの設定(PRINT と共に)	278
SQR	平方根	279
STOP	実行の中断	280
STR$	数式をストリングに	281
STRING$	文字を指定個数並べてストリングに	282
SWAP	変数の値の交換	283
SYSTEM	DOS に戻る	284
TAB	スペースの挿入(PRINT と共に)	285
TAN	正接	286
TIME$ 変数,	時刻の読み取りと設定	287
TIMER	現在の時刻を秒単位で返す	289
TIMER	経過時間ごとの割り込み処理を制御する	290
TRON, TROFF	トレースの開始と中止	291
UNLOCK	レコード保護指定の解除	292
USR	機械語サブルーチンの呼び出し	294
VAL	ストリングから数値に	295

VARPTR …… 変数，ファイル制御ブロックのアドレス …… 298
VARPTR$ …… 変数のアドレスをストリングで返す …… 300
VIEW …… ビューポートの定義 …… 301
VIEW PRINT …… 文字表示領域の定義 …… 305
WAIT …… 実行の保留 …… 306
WHILE, WEND …… 条件が真のときにループ内を実行 …… 308
WIDTH …… 出力行の幅 …… 310
WINDOW …… ウインドーの定義 …… 313
WRITE …… データの画面表示（区切り文字つき） …… 317
WRITE# …… 順次ファイルへの書き込み（区切り文字つき） …… 318

付録A メッセージ集

付録B 参照資料

変数型宣言文字 …… B-1
演算子 …… B-1
ファイル指定とパス名 …… B-3
　ファイル指定 …… B-3
　パス名 …… B-3
装置名とドライブ指定子 …… B-3
　装置名 …… B-3
　ドライブ指定子 …… B-3
画面モード …… B-4
数学関数の公式 …… B-6
キーボードと走査コード …… B-8
数値変換表 …… B-9

第1章

BASICの始動と操作の方法

BASICの始動

BASIC 始動の手順は次のとおりです。

1. システム・ユニットの電源を入れる。
2. DOS ディスケットを入れる。
3. 日付と時刻を入力する。
4. BASIC と入力する。

この操作の詳細は次のとおりです。

日付と時刻の入力

DOS を始動すると日付と時刻の入力要求があります。日付は，年/月/日または年-月-日の形式で入力してください（文字をタイプし ↵ キーを押すことを「入力」と呼びます)。時刻は，時：分：秒の形式で入力します。日付や時刻の入力を省略する場合は，↵ キーのみを押してください。

DOS プロンプト

日付と時刻を入力すると A>_ （ハード・ディスク・システムの場合は C>_）が表示されます。これは DOS が入力待ち状態であることを示す DOS のプロンプトです。英字（A）は，現在の DOS のドライブを表します。

BASIC コマンド

DOS プロンプトで示されたところに BASIC と入力すると，BASIC コマンドが実行され，BASIC の初期画面が表示されます。これは，BASIC が始動したことを意味します。BASIC は DOS ディスケットに入っています。BASIC コマンドには，指定できる選択肢があります。☞ 選択肢の詳細は本章最後の「BASIC 始動時に指定できる項目」。

DOS に戻る

BASIC のモードから DOS に戻るには次のコマンドを入力してください。

```
SYSTEM
```

システム・リセット

DOSを再び始動させるには，前面+Ctrl+削除キーを押します。システム・リセットはBASICのモードでもできます。

システム・リセットは，プログラムのデバック中に記憶域の内容を破壊してしまい，作業を続行できない場合になどに有効です。システム・リセットを行うと，現在の記憶域の内容は無効となり，再びDOSが記憶域に読み込まれます。

ハード・ディスク

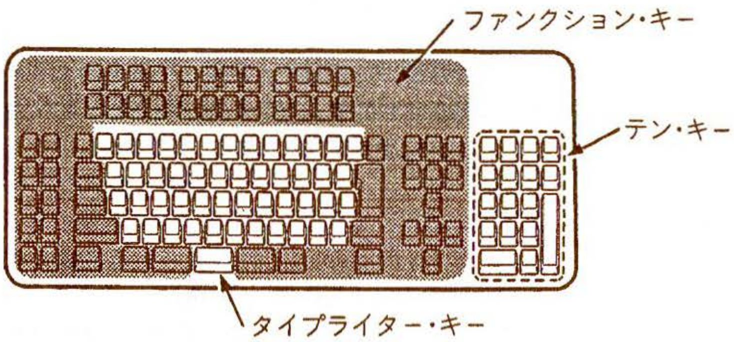
ハード・ディスク・システムの場合，ハード・ディスク初期設定プログラムにより初期設定がなされ，日本語DOSがすでに導入されていなければなりません。この作業手順の詳細は，「ハード・ディスクをお使いいただくために」を参照してください。

キー操作

ここでは，BASICのモードでプログラムを作成し，実行する作業のために必要なキー操作を中心に説明しています。ここで説明しているもの以外のキー操作については，次のマニュアルを参照してください。

システム操作ガイド
日本語DOSユーザーズ・ガイド入門編
日本語DOSユーザーズ・ガイド参照編
日本語文節変換プログラム ユーザーズ・ガイド

キーボードの各キーは用途により次の名前で呼ばれることがあります。



タイプライター・キー データを入力，表示するために使用します。
英数字，ひらがな，カタカナ，特殊文字をタイプできます。

テン・キー 入力モードとは無関係に常に数字入力ができるキーです。

ファンクション・キー キーそれぞれが固有の働きを持っています。カーソル移動キー，入力モード選択キー，プログラム・ファンクション・キー（以降PFキーと呼びます）があります。

本書ではキーボードの「前面キー」を[前面]，「ページ印刷」を[印刷]と記述しています。

文字を入力するキー

英数字の入力

- 英小文字，数字キーの左（左下）側に書かれた特殊文字を入力，表示するためには，[英数]キーを押して離します。画面の入力モード表示行（画面最下行）には英数と表示されます。

  この状態で[⇧]キーを押したままタイプすると，英大文字，キーの上（左上）側に書かれた特殊文字を入力，表示できます。[⇧]キーを離すと元の入力モードに戻ります。

- 英小文字を英大文字に変える場合は，[前面]キーを押したまま[英数]キーを押して離します。画面の入力モード表示行にはCAPSと表示されます。

  この状態で[⇧]キーを押したままタイプすると，英小文字，キーの上側に書かれた特殊文字を入力，表示できます。[⇧]キーを離すと元の入力モードに戻ります。

英数字の場合，半角文字と全角文字を表示できます。

半角文字を表示するには，[半角]キーを押します。半角文字の入力モードになると，入力モード表示行に半角と表示されます。

全角文字を表示するには，[前面]＋[半角]キーを押します。全角文字の入力モードになると，入力モード表示行に全角と表示されます。

この入力モードから抜ける場合は，他の入力モード設定のキーを選択してください。

カタカナの入力

カタカナとキーの下側の特殊文字を表示，入力するためには，カタカナキーを押します。画面の入力モード表示行にはカタカナと表示されます。キーの右側に書かれた文字を入力，表示するには，⇧キーを押します。

カタカナも半角文字と全角文字を表示できます。操作方法は，英数字の場合と同じです。

この入力モードから抜ける場合は，他の入力モード設定のキーを選択してください。

ひらがなの入力

ひらがなとキーの下側の特殊文字を表示，入力するためには，ひらがなキーを押します。画面の入力モード表示行にはひらがなと表示されます。キーの右側に書かれた文字を入力，表示するには，⇧キーを押します。

この入力モードから抜ける場合は，他の入力モード設定のキーを選択してください。

漢字の入力

漢字を入力する方法には次の 3 つの方法があります。

1 かな漢字変換

2 区点コード入力

3 連続コードと内部コード入力

一般的に 2 と 3 の方法は，1 の方法で漢字を表示させることができない場合に，用います。各漢字に割り振られている JIS 区点コード，連続コード，内部コードについては，漢字コード一覧表を参照すると調べることができます。

1　かな漢字変換

日本語 DOS K2.7 以上のバージョンでは，漢字一字ずつを変換していく方式（単漢変換）だけでなく，文節単位に変換していく方式（文節変換）も備えています。したがって，通常のかな漢字変換の操作はこの文節変換で行うことができます（文節変換を行うには，文節変換用の基本辞書や拡張辞書が必要です）。

文節変換位置は，画面上の任意の位置で変換する方式と，25 行目を使用して行う方式の 2 とおりがあります。変換の方法はどちらの場合も同じです。ただし，次の条件のときは画面上の任意の位置で変換する方式では変換できません。

- グラフィック・モードのとき
- LOCATE ステートメントでカーソルを表示しないように指定している場合

これら以外の場合でも，画面上の任意の位置で文節変換できないような状態が起きた場合，自動的に 25 行目で文節変換する方式に切り換わります。

文節変換の手順は次のとおりです。

1　かな漢字変換モードにするために，[漢字]キーを押します。

2　読みをタイプします。

3　[変換]キーを押して該当する漢字を選びます。（該当する漢字が表示されるまで，[変換]キーを何度か押してください。）

文節変換で使用する基本的なキー操作は次のとおりです。

かな漢字変換モードに入る/戻る	[漢字]
変換位置を切り換える	[Ctrl]+[漢字]

かな漢字変換を終了する場合は再び漢字キーを押してください。

文節変換には，このほかにも機能があります。文節変換に関する詳細は，文節変換プログラム・ユーザーズ・ガイドを参照してください。

文節変換用の辞書を使用せずにかな漢字変換を行う場合は漢字一字ずつを変換していく方法（単漢変換）で行います。

単漢変換の手順は次のとおりです。

1 かな漢字変換モードにするために漢字キーを押します。

2 読みをタイプします。

3 変換キーを押すと番号と共に同じ読みの漢字が表示されます。

4 番号を選んでください。

5 一度目の表示の中に該当する漢字が無い場合は，再び変換キーを押してください。

熟語辞書を使用すると，上記単漢変換の方法で熟語単位にかな漢字変換を行うこともできます。

2 区点コード入力

区点コード入力は次のように行います。

1 漢字キーを押して漢字入力モードにします。

2 前面＋漢字キーを押します。区点コード入力のモードになります。

3 JIS区点コードをテン・キーを使用してタイプします。

4 実行キーを押します。漢字が表示されます。

5 区点コード入力モードを終了するためには，ESCキーを押します。

6 漢字入力モードを終了するためには，再び漢字キーを押します。

3　連続コードと内部コード入力

次の操作を行うと対応する文字を表示させることができます。

- 1バイト・コード文字は，[前面]キーを押したまま，テン・キーで3桁の連続コードをタイプし，[前面]キーを離します。

  ただし，連続コード1～6，16，21～23，25の文字は，この方法で表示させることができません。PRINT CHR$（連続コード）の形式を使用してください。

  連続コード29～31の文字をプログラムの実行命令として，画面に出力することはできません。

- 2バイト・コード文字は，[前面]キーを押したまま内部コード（16進数）に相当する10進数を1バイトごとにタイプし，[前面]キーを離します。

  たとえば，漢字の「本」は内部コード967Bです。最初の1バイト目96の10進数は150です。次の1バイト7Bの10進数は123です。この2バイト・コード文字を表示するときは，まず[前面]キーを押したまま150をタイプし一度[前面]キーを離します。次に再び[前面]キーを押したまま123をタイプし[前面]キーを離すと漢字の「本」が表示されます。

これらの操作は，画面の入力モード表示行で表示される入力モードとは，無関係に表示できます。

数値の入力

数値をまとめて入力する場合には，テン・キーを使用すると便利です。テン・キーは，現在の入力モードとは無関係に数値を入力できます。すなわち，仮にひらがな入力のモードであっても数値入力モードに変更せずに数値を入力できます。

JIS 区点コード，連続コードを入力するときは，このテン・キーを使用します。

プログラムを編集するキー

プログラムを入力する

[↵] タイプした表示行をプログラムとして記憶するときは，[↵]キーを押します。

[Ctrl]+[↵] 画面の複数行にわたる長いプログラム行を表示するときは，[挿入] モードで[Ctrl] +[↵] キーを押すとカーソルを置いた位置から画面の右端まで自動的にスペースが挿入されます。プログラム行は半角文字で 255 文字までが有効です。プログラム行として記憶するためには，[↵]キーを押す必要があります。

カーソル移動キー

[↖] カーソルをホーム・ポジション（画面左上隅）へ移動します。

[Clear] 画面を消去し，カーソルをホーム・ポジションへ移動します。

[↑] カーソル上方移動キー。カーソルを真上へ移動します。

[↓] カーソル下方移動キー。カーソルを真下へ移動します。

[←] カーソル左方移動キー。カーソルを左へ移動します。

[→] カーソル右方移動キー。カーソルを右へ移動します。

[Ctrl]+[→] 次の語の先頭へ文字を移動します（語とは，半角の英数

字，ひらがな，カタカナ，漢字から成る文字を言います。特殊文字やスペースは，含みません）。

Ctrl + ← 前の文字の先頭文字へ移動します。

前面 + ↖ 論理行の行末へカーソルを移動します（論理行は，タイプした後 ↵ キーを押すことにより，すでに記憶されている行を言います）。

⇄ 次のタブ位置（半角文字で 8 文字ごと，すなわち 1,9,17,25...番目の文字位置）までカーソルを移動します。ただし，挿入モードのときは自動的に次のタブの位置までスペースを挿入します。

編集キー

Ctrl + ↖ カーソルを置いた位置から行末までを消去します。

ESC カーソルを置いた行を消去します。

Ctrl + ↓ カーソルを置いた位置から行末までと，カーソルより下に表示されているものを消去します。

挿入 挿入モードにします。挿入モードを解除するためには再び 挿入 キーを押してください。またカーソル移動キーや ↵ キーを押した場合も挿入モードは解除されます。

削除 カーソルを置いた文字を削除します。

後退 カーソルの左側の文字を削除し，その位置にカーソルを移動します。

実行を制御するキー

[前面]＋[Pause]　キーを押したときに実行されていた命令の次の命令で，プログラムの実行を中断しコマンド・レベルに戻ります。

[Pause]　プログラムの実行や印刷，またはプログラムの表示を一時停止します。いずれかのキーを押すと実行を再開できます。ただし，次のキーでは，再開できません。

[Pause]　[印刷]　[Ctrl]　[Scroll Lock]　[カタカナ]　[英数]
[⇧]　[前面]　[ひらがな]

[Ctrl]＋[前面]＋[削除]　システム・リセット（DOS の再始動）を行います。

予約語を表示するキー

頻繁に使用する予約語（BASIC が命令を解釈するために決めた言葉）は，[前面]＋[A]～[X] のいずれかを押すことによって，対応する予約語の後ろにスペースを 1 つつけて表示させることができます。入力は [↵] キーを押します。

[A]～[X] に対応する予約語は，次のとおりです。

[A]	AUTO	[N]	NEXT
[B]	BSAVE	[O]	OPEN
[C]	COLOR	[P]	PRINT
[D]	DELETE	[Q]	なし
[E]	ELSE	[R]	RUN
[F]	FOR	[S]	SCREEN
[G]	GOTO	[T]	THEN
[H]	HEX$	[U]	USING

I	INPUT	V	VAL
J	JIS	W	WIDTH
K	KEY	X	XOR
L	LOCATE	Y	なし
M	MID$	Z	なし

印刷の指定を行うキー

印刷　表示画面の内容を印刷します。

前面＋印刷　表示画面の内容を大きく印刷します（グラフィック・モードでのみ有効）。

Ctrl＋印刷　メッセージが表示されるかまたは，↵キーが押されるたびにその表示内容を印刷します。このモードは，エコー印刷といいます。このモードを解除するためには，再びCtrl＋印刷キーを押します。

BASICのプログラム

BASIC のプログラムは次のような形式です。

```
10 '------ 合計を求める ------
20 '                         ↑
30 CLS : SUM = 0          [注 釈]
40 INPUT "値は" ; V          ↓
50 IF V = 0 THEN END  '0が入力されたら終了
60 SUM = SUM + V
70 PRINT "現在までの合計は ";SUM
80 GOTO 40      ↑
↑          [ステートメント]
[行番号]
```

このように BASIC のプログラム行は,

行番号	ステートメント	注 釈

のように記述します。

プログラムを記述するときに, 行番号, 予約語 (プログラム命令として決められた言葉), 変数の前後には, 区切りが必要です。これらは, 通常スペースで区切りますが, 算術演算子 (+, −, *, /, など) や関係演算子 (=, <, > など) は, 変数との間にスペースを入れる必要はありません。また予約語や変数とステートメントの区切り文字「:」の間にもスペースは, 必要ありません。

前述のプログラム例は，次のようにも記述できます（スペースをつめた例です）。

```
10 '------ 合計を求める ------
20 '
30 CLS:SUM=0
40 INPUT "値は";V
50 IF V=0 THEN END  '0が入力されたら終了
60 SUM=SUM+V
70 PRINT "現在までの合計は ";SUM
80 GOTO 40
```

行番号，ステートメント，注釈に関する詳細は，次の各項目を参照してください。

行番号

プログラム行はすべて行番号から始める必要があります。**行番号は 0 から 65529 の範囲内の整数**を記述します。行番号は記憶域内にプログラムが記憶される順序を示すもので，分岐や編集のための参照番号としても使用します。

行番号は，順番に記述されていれば番号と番号の間隔が不統一であってもかまいません。たとえば，5，10，30，35...というような行番号でも，1，2，3...というような行番号でもかまいません。ただし，普通はプログラムを作成している途中で行を挿入することができるように，10，20，30，40...のような行番号を記述します。AUTO コマンドを使用すると行番号を自動的に表示させることができます。また RENUM コマンドを使用すると行番号を付け替えることができます。

ステートメント

BASIC のステートメントには，実行ステートメントと非実行ステートメントがあります。

前述のプログラムの中の行番号 10 と 20，50 のアポストロフィ（'）より後は，注釈です。これらの注釈は，プログラムの中に記述されていても実行されません。このようなステートメントを非実行ステートメントと呼びます。

これに対して，注釈部分以外の記述はすべてプログラム命令で実行可能ステ

ートメントです。これらのステートメントを実行ステートメントと呼びます。

プログラム行は 255 文字以内（半角文字でかぞえて）までが有効です。この 255 文字には最後にタイプする改行文字も含まれます。ただし，改行文字は，表示されません。

1 つのプログラム行に複数のステートメントを記述することもできます。各ステートメントは，次のように「:」で区切ります。

```
30 CLS : SUM = 0
```

ステートメントは，255 文字までであれば 1 プログラム行に複数記述できます。ただし，あまり多用するとプログラムが読みにくくなる恐れがありますので注意してください。

注釈

注釈は，後でプログラムを読むときに内容を理解しやすくするために記述するものです。注釈は次の形式で記述します。

- 1 つのステートメントを注釈文にする場合は，REM ステートメントまたはアポストロフィ（'）を使用して記述します。

  次の 2 つの行は同じ注釈行となります。

  ```
  10 REM '------ 合計を求める ------
  ```

  ```
  10 '------ 合計を求める ------
  ```

- ステートメントの後ろに注釈をつけるときは，アポストロフィ（'）を使用して次のように記述します。

  ```
  50 IF V = 0 THEN END  '0が入力されたら終了
  ```

DATA ステートメントには，アポストロフィー（'）で注釈をつけることができません。

プログラム行に複数のステートメントを記述する場合，注釈は最後のステートメントに記述してください。次のように記述すると，REM 以降のステートメントは無効になります。

```
10 PRINT "ABCD" : REM 文字の表示 : PRINT "あいう"
```

この例では，REM 文字の表示 : PRINT "あいう"がすべて注釈とみなされ，PRINT "あいう"は，実行可能ステートメントとして解釈されません。これは，アポストロフィー（'）を使用した場合も同様です。

```
10 PRINT "ABCD" '文字の表示 : PRINT "あいう"
```

この例の場合もPRINT "あいう"は注釈の一部とみなされます。

プログラムの作成と実行

プログラムの作成，実行，保管，呼び出しなどの一連の作業は，次のように行います。ここで，紹介する各ステートメントの詳細は文法編を参照してください。

プログラムの作成

BASICを始動するとOkの合図が表示されます。この合図は，BASICの用意が整ったことを表しています。この状態をコマンド・レベルといいます。

プログラムは，このコマンド・レベルで作成します。前述のように各行の先頭に行番号をつけてステートメントを記述してください。プログラムを記憶域に読み込むために，1行ごとに[↵]キーを押してください。

プログラムの実行

プログラムを入力後，実行するためにはRUNコマンドを次のように入力してください。

```
RUN
```

プログラムの保管

プログラムをディスクに保管するためにはSAVEコマンドを使用し，たとえばAドライブにPROGという名前で保管するには，次のように入力します。

```
SAVE "A:PROG"
```

A:はAドライブにプログラムを保管する指定です。このダブル・コーテーションで囲まれた部分は，**ファイル指定**といいます。**ファイル指定**に関しては，次章に詳しい説明があります。

プログラムの読み込み

保管したプログラムを再び記憶域に読み込むには，LOADコマンドを使用して次のように入力します。

```
LOAD "A:PROG"
```

リストの表示

作成したプログラム・リストを表示させるには，LIST コマンドを次のように入力してください。

```
LIST
```

プログラムの作成や変更は，このような一連の作業を通して行います。プログラムをリストで表示し，修正を行ったプログラム行に対して［↵］キーを押すことを忘れないでください。［↵］キーを押さないとプログラム行は記憶域に記憶されません。

〈コマンドとステートメント〉

本システムの BASIC のコマンドとステートメントには，特別な使い分けや，制約はありません。コマンドとステートメントのどちらもプログラム内に記述でき，コマンド・レベルでも使用できます。コマンドやステートメントと呼ばれているのは，コマンド群がプログラム編集の作業命令を目的として作成されたためです。「コマンドだから～」または「ステートメントだから～」という使用上の差は特別に無いため，あまり配慮する必要はありません。コマンドをプログラム内で使用する場合は記憶域内に現在存在するプログラム・ファイルが対象となることが多いので注意して指定してください。

プログラム実行のモード

プログラムは，行番号をつけた命令の集合を実行するのが普通ですが，プログラム行の 1 行分をコマンド・レベルでタイプし，実行することもできます。このモードを直接モードと呼び，前者のモードを間接モードと呼びます。

直接モード

直接モードで実行する場合，行番号なしでステートメントをタイプし［↵］キーを押すとタイプした文字が解釈され実行されます。直接モードは，タイプした文字を記憶しておくことはできませんが，一行のみ試すときに便利です（このユーザーズ・ガイドでは，文字をタイプして引続き［↵］キーを押すことを「入力する」と呼びます）。

次のように入力すると画面の最下行から 2 番目の行に表示されているプログラム・ファンクション・キー（以下 PF キーと記述します）の機能表示が消去されます。

```
KEY OFF
```

もう一度表示させるためには次のように入力してください。

```
KEY ON
```

次のように「:」で区切り，複数のステートメントを入力することもできます。

```
FOR I = 1 TO 30 : PRINT "*"; : NEXT I
```

間接モード

プログラムは，各行に行番号を付けた一連の BASIC のステートメントを記述します。プログラムの実行は「プログラムの作成と実行」の項目にあるように RUN コマンドを使用します。

プログラムの修正

BASIC インタープリターでプログラムを作成する場合，BASIC インタープリターが持っているプログラム編集機能を使用して作成します。プログラム編集機能は，コマンド・レベルでタイプした文字を，編集，処理する機能を持っています。

このプログラム編集機能には次のような特徴があります。

- 長いプログラム行をタイプする場合，物理画面の 1 行分を超えても，そのままタイプし続けてください。カーソルは，画面上の次の行で折返します。スペースを埋めてカーソルを次の行に置くには，[挿入]キーを押して挿入モードに入った後[Ctrl]+[↵]キーを押します。

- BASIC のプログラム編集機能は，注釈，DATA ステートメント，引用符で囲まれたストリング以外の文字をすべて大文字に変換します。ただし，タイプするときは大文字でも小文字でも入力できます。

[↵]キーを押すことによって有効となったプログラム行を BASIC は 1 つの文字ストリングの単位として扱います。このような行を論理行と呼びます。

BASIC は実行時に正当な構文か否かのチェックを行ないます。行番号で始まるステートメントを入力しただけでは，チェックされません。ただし，JIS 区点定数と JIS16 進定数の場合は例外です。&K，&J の後ろに無効なコードを入力するとエラーになります。

文字の変更

行をタイプしているときに誤りに気づいた場合は，カーソル移動キーで誤った文字の上までカーソルを動かし，正しい文字をタイプします。その後，[前面]＋[↖]キーを押すとその行の終わりまでカーソルを移動することができます。

たとえば，次のようにタイプしたとします。

```
LOAD "V;PROG
```

間違ってB:のかわりにV;とタイプしてしまいました。英字の場合，[Ctrl]キーを押したままカーソル移動キーを押すと，カーソルを区切り文字で区切られた英字の先頭に移動することができます。したがって，この場合[Ctrl]キーを押したまま2度カーソル移動キーを押すと，カーソルはVの上に移動します。

```
LOAD "V;PROG
```

そしてB:をタイプします。

```
LOAD "B:PROG
```

その後，[前面]＋[↖]キーを押すとカーソルは，現在行の行末に移動します。

文字の削除

文字を削除するためには，[削除]または[後退]キーを使用します。

たとえば次のようにタイプしたとします。

```
DEELETE
```

余分のEを削除するためには，まずカーソルが削除したいEの上にくるまでカーソル移動キーを押します。そして[削除]キーを押します。

```
DEELETE
```

間違った文字が今タイプしたばかりの文字の場合は，［後退］キーを押すとカーソルを1つ戻すとともに文字を削除することができます。たとえば次のようにタイプしたとします。

```
DELETT
```

［後退］キーを押すと次のようになります。

```
DELET
```

文字の挿入

文字を挿入する場合は，文字を挿入したい位置の次の文字位置にカーソルを移動します。［挿入］キーを押して文字をタイプすると文字がカーソル位置に挿入されます。この挿入のモードはカーソル移動キーを使用すると自動的に解除されますが，もう一度［挿入］キーを押して解除することもできます。

たとえば，次のようにタイプしたとします。

```
LIS 10
```

LIST の T を挿入するには，カーソルがスペースの上にくるまでカーソル移動キーを押します。

```
LIS 10
```

そして，［挿入］キーを押したのち T をタイプします。

行の消去

タイプ中の行を消去するときは，［ESC］キーを使用します。行が論理行の場合は論理行が物理画面の複数行にわたるような行であっても，その論理行全体を消去します。

行の追加

すでにあるプログラムの行番号**以外**の行番号を使用し，プログラム行を入力してください。この行は現在記憶域にあるプログラムの一部として記憶されます。このとき既存の行番号を使用してプログラム行を入力すると，既存の行は新しい行で置き換えられます。

現在次のようなプログラムが記憶域にあるとします。

```
10 '------ 合計を求める ------
20 '
30 CLS : SUM = 0
40 INPUT "値は";V
50 SUM = SUM + V
60 PRINT "現在までの合計は ";SUM
70 GOTO 40
```

このプログラムの行番号 40 の次に IF V = 0 THEN END という行を挿入する場合は，このステートメントを次のように記述します。

```
45 IF V = 0 THEN END
```

記憶域のプログラムを LIST コマンドで表示させると次のようなプログラムが表示されます。

```
10 '------ 合計を求める ------
20 '
30 CLS : SUM = 0
40 INPUT "値は";V
45 IF V = 0 THEN END
50 SUM = SUM + V
60 PRINT "現在までの合計は ";SUM
70 GOTO 40
```

挿入するステートメントの行番号は，前後の行番号の間の数であればどの数であってもかまいません。

行番号を付け替える場合は，RENUM コマンドを使用すると便利です。

行の削除

既存のプログラム行を削除するときは，行番号のみを入力します。

複数のプログラム行を削除する場合は，DELETE コマンドを使用すると便利です。

[ESC] キーは，画面から行の表示を消去することはできますが，記憶域にあるプログラム行を削除することはできません。

プログラムの削除

現在記憶域にあるプログラム全体を削除するためには，NEW コマンドを使用します。NEW と入力してください。

NEW は，通常新しいプログラムを入力するときに記憶域を解放するために使用します。

構文エラー

プログラム実行中に構文エラーが発見されると，BASIC はエラーを起こした行を自動的に表示します。たとえば次のように表示します。

```
Ok
10 A=2$12
RUN
Syntax error in 10
Ok
10 A=2$12
```

表示された行で修正を行い ↵ キーを押すと，修正された行がプログラム内に記憶されます。

このように行を修正しプログラムとして記憶させている間，プログラムの実行は，一時中断され，現在実行していたプログラムに対し，主に次のことが行われます。

- 変数や配列の内容は消去されます。これらはゼロまたは空ストリングに設定されます。
- 配列の宣言や変数の型宣言は取り消されます。
- オープンされていたファイルはすべてクローズされます。
- プログラムを続行するために CONT を使用することはできません。

変数の内容を調べる

BASIC インタープリターは，プログラム実行後変数にどのような値が割り当てられているか調べることができます。

これは，構文エラーが起きてプログラムの実行が中断した場合や，実行後の結果が思いどおりでない場合に便利です。

次のプログラムを実行したと仮定します。

```
10 DATA 90,65,75,35,80,40,50,90,95,70,45,60
20 NDATA = 12 : SUM = 0
30 FOR I = 1 TO NDATA
40   READ INDATA
50   SUM = SUM + INDATA
60 NEXT I
70 AV = SUM/NDATA
80 PRINT "平均値は";AV
```

変数SUMの値を調べるには，次のように入力します。

```
?SUM
```

変数SUMの値が表示されます。

ただし，構文エラーが起きてプログラムの実行が中断されたとき，変数の内容を調べるには，行を修正する前にBreakキー機能を使用してコマンド・レベルに戻ってください。実行した直後の変数の内容を調べることができます。変数の内容を変更した後にプログラムを実行する場合は，RUN コマンドを使用してください。変数の内容を変更した場合に CONT コマンドを使用すると正しい結果が得られません。

BASIC始動時に指定できる項目

BASIC を始動するときは BASIC とコマンドを入力しますが，この BASIC コマンドにはコマンド入力時に同時に指定できる項目があります。この項目は，必要に応じて指定する選択肢です。BASIC で初めてプログラムを作成される方はこの選択肢を指定しなくても，十分いろいろなプログラムを作成できますので指定できる選択肢があることだけを確認してください。

選択肢は次の形式で指定します。

BASIC　**ファイル指定**　<**標準入力指定**　>**標準出力指定**　/I

/F: **ファイル数**　/S: **ブロック・サイズ**　/C: **通信バッファー**

/M: **最大作業領域**：**最大ブロック・サイズ**　/D

各選択肢は省略できます。すなわち，指定する必要のあるもののみ指定してください。

注) /I，/F，/S は他の BASIC と互換性を持たせるために構文に入っています。したがって，これらの選択肢は通常ほとんど必要ありません。

ファイル指定

ファイル指定はBASIC始動後すぐに読み込んで実行させるプログラムのファイルを引用符（"）なしのストリング定数で指定します。ファイル指定の形式は，**装置名：**または**ドライブ指定子：ファイル名.エクステンション**のどちらかです。ファイル名は8文字，エクステンションは3文字までです。

<標準入力指定

標準入力指定には，標準入力となるファイル指定をします。形式は**ファイル指定**と同じです。この指定は，省略時の標準入力装置であるキーボードからの入力を変更するものです。この選択肢は/で始まる選択肢より前に置いてください。

>標準出力指定

標準出力指定には，標準出力となるファイルを指定します。形式は**ファイル指定**と同じです。この指定は，省略時の標準出力装置である画面への出力を変更するものです。この選択肢は/で始まる選択肢より前に置いてください。先頭に**もう1つ>をつけると追加して出力**されます。

/I

/Iは，選択肢/Sと/Fを有効にするための指定です。/Iを指定しないと，/Fと/Sに値を設定しても無効になります。本システムのBASICは，ファイル・アクセスに必要な領域を動的に割り振ります。すなわち，要求されたファイル・アクセスに必要な領域をシステムの現状に応じて可能な限り確保しようと試みます。したがって，通常/Fと/Sは使用する必要がありませんが，システムによってはデータを静的に扱うものもあります。/Fと/Sはこのようなシステムと互換性を持たせるためのものです。本システムでは，/Iによって/Fや/Sを指定しなくても十分なファイル・アクセスを行えます。

/F

ファイル数は同時にオープンできるファイルの最大数を指定します。指定できる最大値は15です。/Iを指定して/Fを指定しない場合，同時にオープンできるファイルの最大数は3に設定されます。BASICが同時にオープンできるファイル数はDOSのCONFIG.SYSファイルの構成コマンドFILESの値に依存します。ただし，FILESで指定した数のファイルをすべてオープンできるのではなく，このうちのいくつかはDOSとBASICが使用します（標準入出力やBASICプログラムの読み込み/保管のためです）。この値は5つ前後ですがシステムの状況により異なります。BASICで扱える範囲は最大15であるため，CONFIG.SYSファイルのFILESに20以上の値を指定しても同時にオープンできるのは，15までです。本システムのBASICは通常/I /Fを指定しなくても，CONFIG.SYSファイルに十分

な値が設定されていれば同時に15までは可能な限りファイルをオープンします。

/S　**ブロック・サイズ**はランダム・ファイル用のバッファー・サイズを指定します。最大値は32767バイトです。/Iを指定したときに/Sを指定しないとこの値は128バイトになります。通常，本システムのBASICは可能な限り領域を動的に確保するため，/Sを使用する必要はありません。/Sを指定する場合は/S:512を使用すると効率よくランダム・アクセスできます。

/C　**通信バッファー**は非同期通信のデータ受け取り時バッファーの大きさを指定します（非同期通信には，通信カードが必要です）。最大値は32767バイトです。省略時は256バイトです。送信用として常にこのうちの128バイトが割り振られます。高速回線の場合は/C:1024に指定すると効率良く使用できます。ゼロを指定（/C:0）すると通信カードの使用禁止を意味します。この場合通信バッファー域は確保されません。

/M　**最大作業領域**は作業領域の最大バイト数を指定します。最大値と省略値は64Kバイト（&HFFFF）です。この指定は，作業領域の外に何らかのデータ領域を確保するときに使用します。データ領域を確保するためには，この最大作業領域に64Kバイトより少ない値を設定し，その余りをデータ領域の一部に使用します。

最大ブロック・サイズは，作業空間外（高アドレス側）に設定したデータ領域の大きさをブロック単位でを指定します。1ブロックは16バイトです。この指定は，作業空間外（高アドレス側）に設定したデータ領域を壊さないようにするために指定します。

/D　これは数学関数（ATN，COS，EXP，LOG，SIN，SQR，TAN）を倍精度で計算する指定です。/Dを指定したときはパラメーターも倍精度で指定してください。/Dを省略すると関数は単精度で計算され，倍精度計算のための領域は，使用者のプログラムの領域になります。
ファイル数，ブロック・サイズ，通信バッファー，最大作業領域，最大ブロック・サイズの指定は，10進数，8進数（先頭に&Oが必要），16進数（先頭に&Hが必要）のいずれでも指定できます。

例：

次の例は，プログラムのファイル**給与計算**.BASを読み込んで実行します。このとき，省略値としてランダム・ファイル・バッファーに 128 バイト，通信バッファーに 256 バイト，最大作業領域に 64K バイトが割り振られます。ファイル指定のエクステンションを省略した場合.BASが設定されます。

```
BASIC 在庫管理.BAS
```

次の例は，最大作業領域を32768バイトにする指定です。

```
BASIC /M:32768
```

次の例は，最大作業領域を 16 進数で指定したものです（&H9000 は 36K バイト）。ドライブ A のディスケットにある CHKWRT.TST を読み込み実行します。

```
BASIC A:CHKWRT.TST /M:&H9000
```

次の例は，BASIC を始動後実行するプログラムに MYPROG，標準出力指定に DATA.OUT を指定しています。データ入力は通常のキーボードです。プログラムの中でデータの出力に PRINT を使用すると通常の画面に出力すると共に DATA.OUT に出力します。

```
BASIC MYPROG >DATA.OUT
```

次の例は，BASIC を始動後実行するプログラムに MYPROG，標準入力指定に DATA.IN を指定しています。標準入力指定にファイルを指定すると，キーボードからの代りに指定した DATA.IN から読み取ります。データ出力は，通常の画面です。

```
BASIC MYPROG <DATA.IN
```

次の例は，データを¥販売¥鈴木¥報告書というファイルから読み込みます。データ出力に PRINT を使用すると，画面の代わりに指定した¥販売というファイルに追加して書き出します。

```
BASIC MYPROG <¥販売¥鈴木¥報告書 >>¥販売
```

標準入出力を**標準入力指定**や**標準出力指定**で変更する場合は，次の点に注意してください。

- **標準入力指定**にファイルを指定したとき，Break キー機能が押されたり，エラーが起きるとファイルはすべてクローズされ，実行中のプログラムは終了し，制御は DOS に戻ります。

- **標準入力指定**にファイルを指定しても，プログラム中で入力用に "KYBD:"をファイル指定するとキーボード入力になります。

- **標準出力指定**でファイルを指定しても，プログラム中で出力用に "SCRN:"をファイル指定すると画面に出力します。

- ON KEY (n) ステートメントの使用は，**標準入力指定**とは，無関係です。ON KEY (n) は，キー割り込みを行います。

第2章

プログラミングの基本規則

BASICで使用できる文字

BASICは，英字，数字，カタカナ，ひらがな，漢字，特殊文字を使うことができます。

画面に表示される文字は，半角文字と全角文字があります。半角文字は，横方向が全角文字の半分です。半角文字のコードは1バイトの記憶域を使用して記憶されます。全角文字のコードは2バイトの記憶域を使用して記憶されます。そのため半角文字を1バイト・コード文字，全角文字を2バイト・コード文字とも呼びます。英数字やカタカナの場合，同じ文字でも全角文字と半角文字の内部コード（システム内部で文字データとして確認されるコード）は全く異なります。

BASICのプログラムを記述する場合，予約語，演算子，変数名，数値は，半角の英数字で記述しなければなりません。全角の英数字で予約語を記述することはできません。

カタカナ，ひらがな，漢字，全角の英数字はストリング・データとして扱う部分や注釈文に使用できます。

特殊文字

BASIC では次の特殊文字を使用できます。

文字	名前
	スペース
＝	等号，割り当て記号
＋	正符号，連結記号
−	負記号
＊	アステリスク（星印），乗算記号
/	斜線，除算記号
¥	円記号，整数除算記号
^	脱字記号，べき乗記号
(	左かっこ
)	右かっこ
%	パーセント記号，整数型宣言文字
＃	番号（ポンド記号），倍精度型宣言文字
$	ドル記号，ストリング型宣言文字
！	感嘆符，単精度型宣言文字
&	アンパーサンド
,	カンマ
.	ピリオド，小数点
'	単引用符（アポストロフィ），注釈区切り文字
;	セミコロン
:	コロン，ステートメント分離記号
?	疑問符（PRINT ステートメントの省略形）
＜	不等号（より小）
＞	不等号（より大）
"	2 重引用符，ストリング区切り文字
_	下線

予約語

予約語はBASICの命令（コマンド，ステートメント，関数，演算子）としてあらかじめその意味が決められている単語です。予約語は与えられた命令を識別するものです。変数名として予約語を使用しないように注意してください。予約語の中のFNXXXXXXXXは任意に定義された関数の名前を表します。先頭にFNをつけた10文字までの単語は関数と解釈されます。たとえば，FNUMやFNOTEなどの単語はこれにあたり，変数名として使用しようと思っても実際には任意に定義された関数と見なされますので注意してください。

予約語を記述する場合は，予約語として識別できるようブランクまたは特殊文字で区切る必要があります。各コマンド，ステートメント，関数，変数の具体的な機能については文法編で説明します。

予約語

ABS	AND	ASC
ATN	AUTO	BEEP
BLOAD	BSAVE	CALL
CDBL	CDBL $	CHAIN
CHDIR	CHR $	CINT
CIRCLE	CLEAR	CLOSE
CLS	COLOR	COM
COMMON	CONT	COS
CSNG	CSNG $	CSRLIN
CVD	CVI	CVS
DATA	DATE $	DEF
DEFDBL	DEFINT	DEFSNG
DEFSTR	DELETE	DIM
DRAW	EDIT	ELSE
END	ENVIRON	ENVIRON $
EOF	EQV	ERASE
ERDEV	ERDEV $	ERL
ERR	ERROR	EXP
EXTERR	FIELD	FILES
FIX	FNXXXXXXXX	FOR

FRE	GET	GOSUB
GOTO	HEX$	IF
IMP	INKEY$	INP
INPUT	INPUT#	INPUT$
INPUT¥	INSTR	INT
IOCTL	IOCTL$	JIS$
KEY	KEY$	KILL
KLEN	KPOS	KTN$
LEFT$	LEN	LET
LINE	LIST	LLIST
LOAD	LOC	LOCATE
LOCK	LOF	LOG
LPOS	LPRINT	LSET
MERGE	MID$	MKDIR
MKD$	MKI$	MKS$
MOD	MOTOR	NAME
NEW	NEXT	NOT
OCT$	OFF	ON
OPEN	OPTION	OR
OUT	PAINT	PALETTE
PEEK	PEN	PLAY
PMAP	POINT	POKE
POS	PRESET	PRINT
PRINT#	PSET	PUT
RANDOMIZE	READ	REM
RENUM	RESET	RESTORE
RESUME	RETURN	RIGHT$
RMDIR	RND	RSET
RUN	SAVE	SCREEN
SGN	SHELL	SIN
SOUND	SPACE$	SPC(
SQR	STEP	STICK
STOP	STR$	STRIG
STRING$	SWAP	SYSTEM
TAB(	TAN	THEN

TIME$	TIMER	TO
TROFF	TRON	UNLOCK
USING	USR	VAL
VARPTR	VARPTR$	VIEW
WAIT	WEND	WHILE
WIDTH	WINDOW	WRITE
WRITE#	XOR	

BASICで使用できるデータ

BASIC では次の四角で囲まれたデータを識別して使用することができます(文字イメージのデータをストリングと呼びます)。

BASIC で使用できるデータ

- ストリング
- 数 値
  - 整 数
  - 実 数
    - 単精度数
    - 倍精度数

したがって,BASIC では次のデータ型を使用します。

- ストリング
- 整 数
- 単精度数
- 倍精度数

これらのデータ型はそれぞれ記憶域に格納される方法が異なります。データは処理内容に適したデータ型で表現する必要があります。

定数

ストリング定数

2 重引用符で囲んだ 255 文字（半角でかぞえて）までの文字列です。

【例】

```
"ストリング定数"
"¥25,000"
```

数値定数

正または負の数値です（正符号＋は省略できます）。数値定数にカンマを使うことはできません。数値定数は間にスペースが含まれていてもすべて無視されます。数値定数には以下の 7 つの形式があります。

整数

－ 32768 以上＋ 32767 以下の整数。

【例】

```
5278
-327
```

固定小数点数

正または負の実数。

【例】

```
532.25
-35.7
```

浮動小数点数

指数形式で表された正または負の数。浮動小数点数は次の形式をとります。

単精度 ..... **仮数部 E 指数部**
倍精度 ..... **仮数部 D 指数部**

仮数部 ：符号付き整数または固定小数点数（正符号は省略可）
指数部 ：符号付き整数（正符号は省略可）
E，D ：浮動小数点定数

数字を浮動小数点数として表示できる範囲は絶対値が 2.94E-39 ～ 1.7E + 38 までです。

【例 1】
```
23E-2
```

23 は仮数部，-2 は指数部で，23 に 10 の-2 乗をかけるという意味になります。固定小数点で表すと 0.23 です。

【例 2】
```
235.988E-7
```

これを固定小数点で表すと 0.0000235988 です。

【例 3】
```
2359D6
```

これを固定小数点で表すと 2359000000 です。

16 進数

先頭に**&H** をつけた 4 桁までの 16 進数値。16 進数の各桁は 0 ～ F で構成されます。

【例】
```
&H76
&H32F
```

8 進数

先頭に&O をつけた 6 桁までの 8 進数値。8 進数の各桁は 0 ～ 7 で構成されます。

【例】
&O347

JIS16 進定数

JIS16 進コードの先頭に&J をつけたもの。JIS16 進定数は&J の後ろに書かれた JIS16 進コードに対応する連続コードの値を持ちます。JIS16 進数コードは 4 桁の 16 進数で表され，その範囲は上位 2 桁と下位 2 桁がそれぞれ 21 ～ 7E の範囲でなければなりません。範囲外のコードを入力すると Illegal function call（違法関数呼び出し）エラーになります。

【例】
&J2121

これは連続コード 256 の値を持っています。VAL("&J"+"2121")は同じ値になります。

JIS 区点定数

JIS 区点コードの先頭に&K をつけたもの。JIS 区点定数は&K の後ろに書かれた JIS 区点コードに対応する連続コードの値を持ちます。JIS 区点コードは 4 桁の 10 進数で表され，その範囲は上位 2 桁と下位 2 桁がそれぞれ 01 ～ 94 の範囲になければなりません。指定できる JIS 区点定数は JIS 区点コードの範囲に限ります。範囲外のコードを入力すると Illegal function call（違法関数呼び出し）エラーになります。

【例】
&K0101

これは連続コード 256 の値を持っています。VAL("&K"+"0101")は同じ値になります。

変数

変数には次の 2 つの型があります。

- **数値変数**
- **ストリング変数**

変数に定数，計算結果，データ入力ステートメントの結果を割り当てるときは，変数と割り当てるデータ型とが一致していなければなりません。

値を割り当てる前に変数を使用すると，数値変数はゼロ，ストリング変数は空ストリング（長さゼロ）とみなされます。

各変数データが占める記憶空間のバイト数は次のとおりです。

変数型	記憶領域のバイト数
整変数	2 バイト
単精度変数	4 バイト
倍精度変数	8 バイト
ストリング変数	可変（0 ～ 255）

ストリング変数の長さは可変で 0 ～ 255 バイトの範囲です。変数の長さは変数に割り当てられるストリングの値の長さによって決まります。

変数名

BASICの変数名は最大40文字です。

変数名に使用する文字は，**英字，数字（いずれも半角のみ），小数点**です。ただし，**最初の文字は英字でなければなりません。**この他に，変数型を識別するために（$，%，!，#）を名前の最後の文字として使用します。変数名を決める場合は次の点に注意してください。

- 変数名に漢字，ひらがな，カタカナを使用することはできません。
- 変数名に予約語を使用することはできません。型宣言文字（$，%，!，#）の1つを予約語の最後につけたものも無効です。名前の中に予約語を含めることはできます。

【例】
EXP=5やABS=10は，EXPやABSが予約語であるため無効となります。
EXPONENT=5は，有効です。

変数型

変数型は変数名によって決まります。変数名は変数名の最後に次の型宣言文字をつけて変数型を表します。**変数名の最後に型宣言文字のついていない変数は，単精度変数とみなされます。**

変数型	型宣言文字
整変数	%
単精度変数	!
倍精度変数	#
ストリング変数	$

【例】
整変数 .... LIMIT%, NUM%
単精度変数 .... MINIMUM!, ABC
倍精度変数 .... PI#
ストリング変数 .... N$, INDATA$

変数型を宣言する方法には DEF 型ステートメントを使用する方法もあります。DEF 型ステートメントは変数名で使用する英字の範囲を定義することによって，まとめて型宣言することができます。

数値精度

数値は次のいずれかの形式で記憶されます。

- **整数**
- **単精度数**
- **倍精度数**

整数は 2 バイトの記憶域に記憶され，その精度は完全です。16 進数，8 進数，JIS16 進定数，JIS 区点定数の形式で入力された定数は，2 バイトの記憶域に記憶され整数として扱われます。

単精度数とは 6 桁の精度を持つ実数です。単精度数は 7 桁で記憶され，7 桁まで表示されます。単精度定数は次のいずれかの形で書かれたものです。

- 7 桁以下の数
- E を使用した浮動小数点形式
- 最後に ! （感嘆符）をつけた数

【例】

```
46.8
-1.09E-06
3489.0
22.5!
```

倍精度数とは 16 桁の精度を持つ実数です。倍精度数は 17 桁で記憶され 17 桁まで表示されます。倍精度定数は次のいずれかの形で書かれたものです。

- 8 桁以上の数
- D を使用した浮動小数点形式
- 最後に#（番号記号）をつけた数

【例】

```
345692811
-1.09432D-06
3289.0#
7654321.1234
```

整数，単精度数，倍精度数の範囲と精度をまとめると次のようになります。

数値型	範囲	精度
整数	-32768～32767	完全
単精度数	2.94E-38～1.7E+38	6桁
倍精度数	2.94D-38～1.7D+38	16桁

数値精度の変換

BASICの数値変換には次の規則があります。

1 数値が異なる数値変数で割り当てられた場合，その数値は割り当て先の変数名で宣言された精度で記憶されます。

【例】

```
10 A%=23.42     '実数を整数変数に割り当てる
20 PRINT A%
RUN
 23
```

2 高い精度の値を低い精度の変数に割り当てた場合（たとえば，倍精度を単精度に），値は四捨五入された低い精度の値として表示されます。

【例】

```
10 C=55.8834567#
20 PRINT C
RUN
 55.88346
```

これは，割り当てを行っているステートメントだけでなく，関数やステートメントの評価のときにも起こります。

【例】

A(1.5)とA(2)は同じになります。
X=11.5 MOD 4のXは0になります。

3 低い精度の数を高い精度の数に変換しても，より正確な数値にはなりません。次の式を参照してください。

```
B#=A
```

B#は倍精度変数，Aは単精度変数です。Aは，6桁までの精度なのでB#も最初の6桁までの精度しかありません。

次式は単精度数を倍精度数に割り当てた場合の誤差範囲を表したものです。

ABS (B#-A) <6.3E-8*A

すなわち，表示された倍精度ともとの単精度の差の絶対値は，6.3E-8 にもとの単精度値をかけたものより小さくなります。

【例】

```
10 A=2.04
20 B#=A
30 PRINT A;B#
RUN
 2.04  2.039999961853027
```

4 式の値を求めるときは，算術演算や関係演算の全オペランドが同じ精度（最も高い精度のオペランドと同じ精度）に変換されます。また，算術演算の結果も式のオペランドの中の最も高い精度で返ります。

【例 1】

次の演算は倍精度で行われ，結果も倍精度値で返されます。

```
10 D#=6#/7
20 PRINT D#
RUN
 .8571428571428571
```

【例 2】

この演算は倍精度で行われますが，結果は D（単精度変数）に四捨五入して返えされ，単精度として表示されます。

```
10 D=6#/7
20 PRINT D
RUN
.8571429
```

5 論理演算子は，オペランドを整数に変換し，整数の結果を返します。オペランドの範囲は-32768 ～ 32767 です。範囲外の指定は Overflow（桁あふれ）エラーになります。

配列

配列とは，いくつかの値を集めたグループまたはテーブルのことです。配列は，固有の変数名で参照します。配列に含まれる個々の値を要素と呼びます。ここではBASICにおける配列の使い方について説明します。

配列名や型，配列中の要素の数や配置を設定することを**配列の定義**または**次元宣言**と呼びます。これは，DIMステートメントを使用して行います。たとえば，

```
10 DIM B$(5)
```

と記述した場合，B＄という配列名の一次元配列が作られその要素はみな可変長のストリングで，初期値は空ストリングになります。図で表すと次のようになります。

B$ (0)
B$ (1)
B$ (2)
B$ (3)
B$ (4)
B$ (5)

次の配列宣言は2次元配列の場合の例です。

```
20 DIM A(2,3)
```

この場合は，Aという名前の二次元配列が作られ，名前の中に型宣言文字がないので，配列要素はすべて単精度値です。配列要素の初期値は0（ゼロ）です。図で表すと次のようになります。

列

行				
	A (0,0)	A (0,1)	A (0,2)	A (0,3)
	A (1,0)	A (1,1)	A (1,2)	A (1,3)
	A (2,0)	A (2,1)	A (2,2)	A (2,3)

配列要素数は，配列名に数字の添字をつけて表します。配列の次元数は添字の数によって表します。添字は，配列中の要素の位置を示します。OPTION BASEステートメントを使用して変更しない限り，添字の最も低い位置はゼロです。配列次元数の最大値は255です。次元ごとの要素数の最大値は，次のとおりです。

整数型配列とストリング型配列 32767
単精度実数型配列 16383
倍精度実数型配列 8191

配列を定義する前に，配列要素を使用すると添字の最大値は10に宣言されたとみなされます。たとえば，配列宣言なしに次のステートメントを記述したとします。

```
50 ARY(3)=500
```

この配列はARY（0）～ARY（10）までの要素が11の1次元配列として設定されます。このように配列宣言を省略してもDIM ARY（10）と宣言したことと同じ意味になるため，この後配列宣言を行うと，Duplicate Definition（二重定義）エラーになります。

一度配列宣言した配列の領域を消去するにはERASEステートメントを使用します。ERASEステートメントで消去すれば，再び同じ名前で配列宣言できます。

次のプログラムはDuplicate Definition（二重定義）エラーが発生した例です。

```
10 A(5)=12
20 DIM A(100)
30 PRINT A(5)
40 END
RUN
Duplicate Definition in 20
```

次のプログラムは配列の使用例です。

```
10 DIM YEARS(3,4)
20 YEARS(2,3)=99
30 FOR R = 0 TO 3
40   FOR C = 0 TO 4
50     PRINT YEARS(R,C);
60   NEXT C
70   PRINT
80 NEXT R
RUN
 0 0 0 0 0
 0 0 0 0 0
 0 0 0 990
 0 0 0 0 0
```

数式の演算子

数式に使用できる演算子は次の 3 つです。

- 算術演算子
- 関係演算子
- 論理演算子

その他，BASIC には数値演算のために SQR（平方根），SIN（正弦）などのシステムに常駐する組み込み関数があります。また，組み込み関数にない関数は，DEF FN ステートメントを使用して定義することができます。

算術演算子

算術演算は，次の特殊記号を使用しています。演算の優先準備は，以下の順です。

演算子	演算	例
^	ベキ乗	X ^ Y
−	負の記号	− X
*,/	乗算，浮動小数点除算	X * Y,X/Y
¥	整数除算	X¥Y
MOD	剰余の計算	X MOD Y

整数除算および剰余の計算は次のように行います。

整数除算

整数除算は円記号（¥）を使います。オペランドは除算の前に整数（-32768 ～ 32767）に四捨五入されます。商は切り捨てにより整数になります。

【例】

```
10 A=10¥4
20 B=20.68¥6.99
30 C=20.47¥6.99
40 PRINT A;B;C
RUN
 2 3 2
```

剰余の計算

剰余の計算は，演算子 MOD を使用します。MOD は，整数除算の剰余である整数値を与えます。

【例 1】

```
10 A=7 MOD 4
20 PRINT A
RUN
 3
```

7/4 は商が 1 で剰余が 3 になります。

【例 2】

```
PRINT 25.68 MOD 6.99
 5
```

MOD は整数除算であるため与えた数値は四捨五入して整数値として扱われます。したがってこの計算は，26/7 で剰余が 5 になります。

関係演算子

2つの値を比較するときは，関係演算子を用います。比較は，両方とも数値であるか，両方ともストリングでなければなりません。比較の結果は「真」（-1）か「偽」（0）のどちらかです。次の表は，演算子とその意味です。

演算子	意味	式の例
＝	等しい	X=Y
＜＞または＞＜	等しくない	X＜＞Y,X＞＜Y
＜	より小さい	X＜Y
＞	より大きい	X＞Y
＜= または =＜	以下である	X＜=Y,X=＜Y
＞= または =＞	以上である	X＞=Y,X=＞Y

等号は変数に割り当てるときにも使用します。

数値比較

算術演算子と関係演算子を1つの式の中で組み合わせて使用している場合，常に算術演算の方が先に行われます。

【例 1】

```
X+Y<(T-1)/Z
```

（X + Y）の値が（T-1）/Z の値より小さいとき，真（-1）になります。

【例 2】

```
10 X=100
20 IF X<>200 THEN PRINT "等しくない" ELSE PRINT "等しい"
RUN
等しくない
```

この例の関係式の値は真になるため（100は200に等しくない），IFステートメントのTHENの部分が実行されます。

【例3】

```
PRINT 5<2;5<10
 0 -1
```

この例の最初の結果は5が2より小さくないので偽(ゼロ)になります。次の式は，5が10より小さいので真（-1）になります。

ストリング比較

ストリング比較では，一方の文字がもう一方の文字より文字の順番が前にあれば，「より小さい」とみなされます。文字の順番は，英文字なら「アルファベット順」，カタカナなら「アイウエオ順」，漢字ならJIS情報交換用漢字符号系の「区点番号順」です。小文字は対応する大文字「より大きい」とみなされます。また文字の順番は，小さい方から数字，英大文字，英小文字，カタカナ，漢字符号系の順になっています。したがって，連続コードの大きさの順が文字の順番を表します。

2つのストリングを実際に比較するときは，各ストリングから同時に1文字をとり，それらの内部コードを比較します(内部コードは連続コードを16進数にしたものです)。全部の内部コードが同じの場合それらのストリングは同じとみなされます。それ以外の場合，内部コードが違った時点で，低いコード番号を持ったストリングは，高いコード番号をもったストリングより小さいとみなされます。ストリングを比較している途中で一方のストリングが終わりになった場合，短い方のストリングがより小さいとみなされます。

先行スペースや後続スペースも有効です。次の関係式はみな真（-1）になります（ストリング定数は二重引用符で囲む必要があります）。

```
"AA" < "AB"
"X&" > "X#"
"Kg" > "KG"
"RICH" < "RICHB"
B$ < "718"          (B$="12543")
```

論理演算子

論理演算子は数値に対して論理演算（ブール演算）を行います。関係演算子が，プログラムの流れに関する判断に使用されるのと同じように，論理演算子も 2 つ以上の関係を調べ，真または偽の値を得るために使用します。論理演算子はビットごとに演算します。

BASIC には次の論理演算子があります（演算子は優先順位の高い順にあげてあります)。

- NOT
- AND
- OR
- XOR
- EQV
- IMP

各演算子が返す結果は次ページのとおりです。「T」は真で-1 の値を持ち「F」は偽で 0 の値を持ちます。演算子は優先順位の高い順にあげてあります。

NOT	X	NOT X
	T	F
	F	T

AND	X	Y	X AND Y
	T	T	T
	T	F	F
	F	T	F
	F	F	F

OR	X	Y	X OR Y
	T	T	T
	T	F	T
	F	T	T
	F	F	F

XOR	X	Y	X XOR Y
	T	T	F
	T	F	T
	F	T	T
	F	F	F

EQV	X	Y	X EQV Y
	T	T	T
	T	F	F
	F	T	F
	F	F	T

IMP	X	Y	X IMP Y
	T	T	T
	T	F	F
	F	T	T
	F	F	T

【例 1】

```
IF A>60 AND B<20 THEN 1000
```

変数Aの値が60より大で，かつBの値が20より小さい場合，結果は真になり，プログラムは行番号1000へ分岐します。

【例 2】

```
IF I>10 OR K<0 THEN 50
```

Iが10より大きいかKが0より小さいとき，または，その両方のとき結果は真になりプログラムは行番号50へ分岐します。

【例 3】

```
50 IF NOT (P=-1) THEN 100
```

Pが-1に等しくないとき，プログラムは100の行に分岐します。NOT (P=-1) は，NOT Pと同じ結果にならないため注意してください。

【例 4】

```
100 FLAG%=NOT FLAG%
```

これは，FLAG%が0または-1の値を持っているときに値を真から偽，または偽から真へ変換します。

論理演算子の働き

演算は次の順序で行われます。

- オペランドが-32768～32767の範囲の整数に変換されます（オペランドがこの範囲にないとOverflow（桁あふれ）エラーになります）。
- 各オペランドを16ビットの列で表すために，負の場合は2の補数をとります（ビットの1と0を反転し，1を加えた数値です）。
- 2つのオペランドの対応するビットごとに，論理演算を行い結果のビットを決定します。

論理演算子は特定のビット・パターンを調べるために使用することもできます。

たとえば，特定位置のビットがオン（1）であるか否かを調べるためにはANDを使用すると便利です。次の例を参照してください。

```
A = 63 AND 16
```

このAには16が設定されます。このビットの演算を2進数で表現すると次のようになります。

	0000000000111111........63
AND	0000000000100000........16
結果	0000000000100000........16

ANDは演算する2つのビットが1のときにのみ1の結果を返します。

このように調べたいビットの位置がオン（1）である値をANDの片側に設定し，内容を調べる可変の値をもう一方に設定することによって，ビット列の内容を調べることができます。

ビット列のすべての桁を調べる場合は-1を使用すると便利です。これは，-1の2の補数が次のようになるためです。

```
B = -1 AND 8
```

	1111111111111111........-1
AND	0000000000001000........ 8
結果	0000000000001000........ 8

Bには8が設定されます。

オペランドのオン（1）条件を両方とも得る場合は OR を使用します。OR はどちらか一方がオン（1）であればオン（1）を返します。次の例を 2 進数表現とともに参照してください。

```
C = 4 OR 2
```

	0000000000000100........ 4
OR	0000000000000010........ 2
結果	0000000000000110........ 6

C には 6 が返ります。

ある数（X）の 2 の補数は次の例で求めることができます。

```
X = 4
T = (NOT X) + 1
```

T には 2 の補数（この場合は-4）が返ります。この演算を 2 進数で表現すると次のようになります。

NOT	0000000000000100........ 4
結果	1111111111111011........-5

	1111111111111011........-5
+	0000000000000001........ 1
結果	1111111111111100........-4

実行順序

算術演算子，関係演算子，論理演算子，数値関数がプログラムに混在する場合，その実行順序は次のようになります。

1 数値関数の呼び出しは最初に行われます。

2 次に算術演算が以下の順に行われます。

^
－（負符号としての－）
＊
￥
MOD
＋,－

3 次に関係演算が行われます。

4 論理演算は最後に以下の順で行われます。

NOT
AND
OR
XOR
EQV
IMP

同じレベルの演算は左から右の順に行われます。演算順序を変更するには，かっこを使用します。かっこ内の演算は最初に行われます。かっこ内においては，通常の演算順序が守られます。演算子が 2 つ続くときは，かっこで区切る必要があります。

数学で記述する代数式とBASICで記述する式の違いを例で表わすと次のようになります。

代数式	BASICの式
$X+2Y$	X + Y * 2
$X-\frac{Y}{Z}$	X - Y/Z
$\frac{XY}{Z}$	X * Y/Z
$\frac{X+Y}{Z}$	(X+Y)/Z
$(X^2)^Y$	(X^2)^Y
X^{Y^Z}	X^(Y^Z)
$X(-Y)$	X * (-Y)

演算子が2つ続くときは，X＊（-Y）の例で示したように，かっこで区切る必要があります。

ストリングの演算子

ストリングの操作には，

- 連結演算子

を使用できます。関係演算子＝，＜＞，＜，＞，＜=，＞=は，2つのストリングを比較するために使用できますが，結果としてストリングではなく数値が返されるため，「ストリング演算子」とはみなされません。関係演算子に関しては，前述の「関係演算子」を参照してください。

BASICにはストリングの操作を行うためのストリング関数があります。ストリング関数は，結果としてストリングを返す点が数値関数と異なります。ストリング関数にはストリング内の一部を返すMID＄や，指定された連続コードに対応する文字を返すCHR＄などの，システムに常駐する組み込み関数を持っています。

連結演算子

正符号（＋）を使用するとストリングを連結することができます。

【例】

```
10 A$="III"
20 B$="XX"
30 PRINT A$+B$
40 PRINT B$+A$
50 PRINT B$+A$+"CCCCC"
RUN
IIIXX
XXIII
XXIIICCCCC
```

第3章

BASICの入出力指定

ファイル

プログラムやデータなどの情報の集合は**ファイル**と呼ばれます。ファイルはディスク（ディスケットまたはハード・ディスク）に保管します。ファイルを入出力する場合には，どの装置のどこのファイルを扱うのかを指定する必要があります。これを**ファイル指定**と呼びます。

BASIC は入出力装置の型にとらわれずに，同じ一定の規則に沿って入出力することができます。すなわち，ディスク・ファイル，画面，キーボード，プリンター，他のコンピューターとの通信などすべて同様の形式で入出力を行うことができます。

ファイル番号

BASIC のプログラムではファイル番号で各ファイルに対する一連の処理を識別します。ファイル番号はファイルのオープン時に 1 ～ 255 の範囲の任意の数値を指定します。

ファイル指定とパス

ファイル指定

ファイル指定は次のようなストリング式で行います。

装置名：

または

ドライブ指定子：ファイル名

ファイル指定は上記のどちらかの形式で指定します。装置名はファイル指定において有効な装置の名前です。ドライブ指定子はディレクトリーまたはファイルのある場所を示し，ファイル名はそのドライブ上，またはディレクトリーの中のどのファイルを探すかを指定します。装置名とドライブ指定子については，後述の装置名表を参照してください。またディレクトリーとは，ファイルの登録簿です。詳細は後述のディレクトリーを参照してください。

装置名またはドライブ指定子を省略した場合，DOSの省略時ドライブが参照されます（DOSの省略時ドライブとは，現行ドライブのことであり，BASICを始動したときDOSプロンプトに表示されていたドライブです）。したがって，このドライブに対する指定の場合には，ドライブ指定子を指定する必要はありません。

ファイル入出力操作のために使用するコマンド，ステートメントは次のとおりです。

BLOAD	BSAVE	CHAIN	FILES
KILL	LOAD	MERGE	NAME
OPEN	RUN	SAVE	

装置名やドライブ指定子を指定する場合には最後に必ずコロン（:）を付けます。またファイル指定をストリング定数で行う場合は，二重引用符で囲む必要があります。

【例】

```
LOAD "A:PROG.BAS"
SAVE "C:在庫.BAS"
```

パス名

パス名は，次のようなストリング式で指定します。

ドライブ指定子：パス名

パス名はディレクトリー名を順に並べたものです。ディレクトリー名とディレクトリー名の間は円記号（￥）で区切ります。円記号（￥）も含めて63文字以下のストリング式で指定します。パス名はファイルが入っているディレクトリーの経路を示します。パス名を指定する変わりに現行ディレクトリーは"."，1レベル上位のディレクトリーは".."と指定することもできます。パス名の指定においても現行ドライブに対する指定の場合，ドライブ指定子を必ず指定する必要はありません。ファイル指定の場合と同様にドライブ指定子だけを指定する場合は，必ずコロン（:）を付けます。

パス名の指定は，ファイル指定で紹介したコマンド，ステートメントで指定できます。また，ディレクトリーの操作は次のコマンドで行います。

CHDIR　　MKDIR　　RMDIR

ファイル指定の場合と同様パス名をストリング定数で行う場合は，二重引用符で囲む必要があります。

【例】

```
CHDIR "B:¥販売¥鈴木¥報告書"
CHDIR "A:¥WORK¥PROG"
```

ファイル名が含まれる場合，最後のディレクトリーとファイル名の間に円記号を入れます。パス名が円記号で始まるとき，BASICはルート・ディレクトリーから探し始めます。その他は，現行ディレクトリーから探し始めます（ディレクトリーに関する詳細は，後述のディレクトリーを参照してください）。

ファイルが現行ディレクトリーに無い場合，現行ディレクトリーからそのファイルにたどるパス名を指定しなければなりません。パスはルート・ディレクトリーから始めるものと，現行ディレクトリーから始めるもののどちらかを指定します。

装置名

装置名は最大 4 文字までの文字の末尾にコロン（:）をつけて表します。各装置を示す装置名は，次のとおりです。

KYBD:	キーボード。入力のみ。
SCRN:	画面。出力のみ。
LPT1:	第 1 プリンター。順次出力またはランダム出力。
COM1:	第 1 非同期通信カード。入力または出力。
COM2:	第 2 非同期通信カード。入力または出力。

記憶装置<ドライブ指定子>

A:	ディスケット・ドライブ。入力または出力。
B:	ディスケット・ドライブ。入力または出力。
C:	ディスケット・ドライブまたはハード・ディスク。入力または出力。

DOS の構成ファイル CONFIG.SYS に文字装置を設定した場合や，DOS コマンドまたはネットワークにおいてドライブが設定された場合はこれ以降のドライブが割り振られます。詳細は各ユーザーズ・ガイドを参照してください。

注） ディスケット・ドライブが 1 台のシステムでドライブを使用するときには，DOS の省略時ドライブを使用してください。たとえば，ドライブ B: で操作しているとき，A: を指定すると，「ディスケットをドライブ A: に差し込み，キーをどれか押してください」というメッセージが出ます。この場合，キーを押すとプログラムを続行することは可能ですが，画面の 24 行目のキー表示が DOS によって壊されるときがあります。24 行目が壊されたときは，CLS を実行するか，Clear キーを押してキー表示をもとに戻してください。

ファイル名

ファイル名は次の形式でファイルに名前を付けます。

ファイル名.エクステンション

ファイル名 .... 1文字～8文字
エクステンション ...... 1文字～3文字

ひらがな，漢字などの2バイト・コード文字は，1文字で2文字と数えます。

ファイル名とエクステンションに使用できるのは次の文字です。

- 次の1バイト・コード文字

  英字
  数字
  !#$%$'()−@^_{}
  カタカナ
  。「」，・゛゜

- すべての2バイト・コード文字

【例】

```
TEMP.DAT
PROG1.BAS
LINE
$$@(%).123
売上表.4月
```

BASIC の場合エクステンションを省略すると.BAS が仮定されます。たとえば，次の SAVE コマンドは，ファイル名を PROG.BAS として DOS の省略時ドライブに保管します。

```
SAVE "PROG"
```

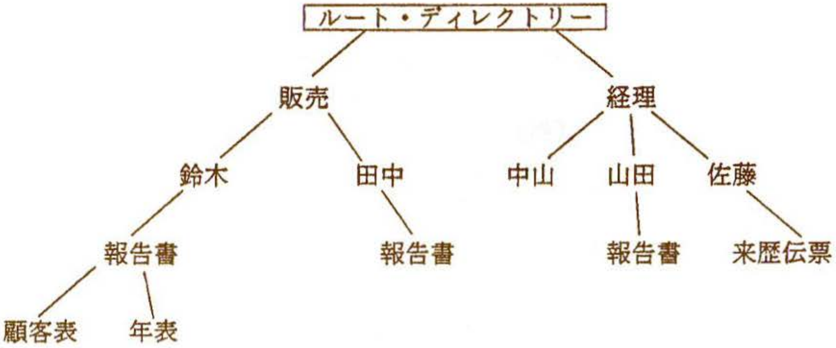
ディレクトリー

ディレクトリーは，ファイルを効率良く管理するために情報を蓄えたファイルの登録簿です。

ディレクトリーの構造

BASIC では，ディスクのファイル管理を容易に行うために，「木構造」のディレクトリーやファイルを形成することができます。

この「木構造」とは，次のような形で管理されるファイル管理形式の一手法です。



上図は，1つのディスクを販売部と経理部で共同使用している場合の例です。このように目的，または 使用者別ディレクトリーを設けると簡単にファイルを見付けることができます。また異なるディレクトリーの下であれば 1つのディスクの中に同じファイル名があっても管理できます。

ディレクトリーのタイプ

新しいディスクをフォーマットしたときは，ディレクトリーが 1 つ作成されます。これをルート・ディレクトリーといいます。たとえば 720KB（80 トラック 9 セクター両面）ディスケットの場合，ルート・ディレクトリーには 112 のファイルが登録できます。

ディレクトリーにはファイルだけでなく他のディレクトリーを登録することができます。このようなルート・ディレクトリー以外のディレクトリーをサブ・ディレクトリーと呼びます。サブ・ディレクトリーはそれ自体がファイルであり，多くの追加ファイルを登録できます。サブ・ディレクトリーに登録できるファイルの数の限界はディスクのスペースに依存します。

サブ・ディレクトリーの命名の仕方およびそれに使用できる文字の種類はファイルの場合と全く同様です。つまりサブ・ディレクトリー名は 1 〜 8 文字の名前の他に 1 〜 3 文字のエクステンションをつけることもできます。ディレクトリー名やファイル名はそれらを管理する上位のディレクトリーが違っていれば，同じ名前のものが重複しても問題ありません。たとえば，前述の図の場合，販売というサブ・ディレクトリーに田中というサブ・ディレクトリーがあるのと同じように，経理というサブ・ディレクトリーに田中というサブ・ディレクトリーがあってもかまいません。同様に販売というサブ・ディレクトリーが佐藤というサブ・ディレクトリーを持つこともできます。

現行ディレクトリー

DOS はあるディレクトリーを一度指定すると，パス名の指定が省略されたときには，そのディレクトリーが指定されたとみなします。このディレクトリーは現行ディレクトリーと呼ばれます。現行ディレクトリーは CHDIR コマンドで指定，変更することができます。また，ディレクトリーの設定は，MKDIR コマンドで行います。ディレクトリーを削除する場合は，RMDIR コマンドを使用します。これらのコマンドの詳細は文法編を参照してください。

画面の使用

画面には，文字を表示するための文字モードと，図形と文字の両方を表示するためのグラフィック・モードがあります。文字モードは，文字に明滅，反転，下線づけなどの属性を指定できます。グラフィック・モードは，LINE，DRAW，CIRCLE，PAINT などのグラフィック・ステートメントで図形を描くことや領域を塗ることができます。この**画面モードは SCREEN ステートメントで指定します**。BASIC 始動時の画面は文字モードです。したがって，文字モード画面として使用する場合は始動時のまま使用できますが，**グラフィック・モードとして使用する場合は，どのグラフィック・ステートメントを記述するより前に SCREEN ステートメントで画面を設定しておく必要があります**。またグラフィック・モードから文字モードに切り換える場合も，再び SCREEN ステートメントで画面モードを設定します。文字属性や文字/図形に色を指定するためには，COLOR ステートメントを使用します。

画面に表示する文字の大きさや個数，図形の座標系は使用している BASIC や画面モードにより異なります。画面は画面上に表示される文字や図形とそれらのまわりの部分から構成されていますが，この文字や図形を**前景**，まわりを**背景**と呼ぶことがあります。

画面モード

画面モードは，使用する BASIC によって異なります。

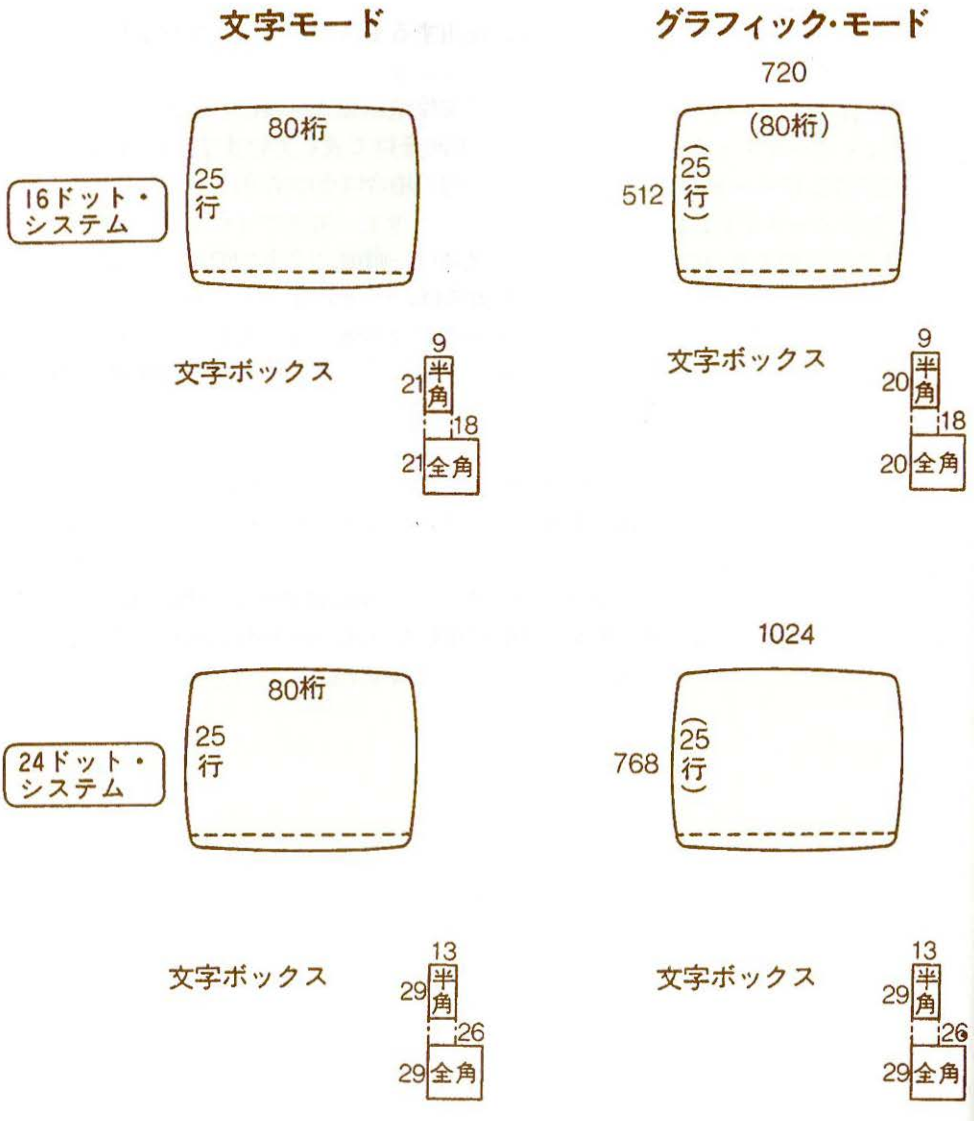
画面に表示される文字や座標（点の数）は次の図のようです。画面の文字は X 方向を桁，Y 方向を行で表しています。桁は文字が半角の場合の桁数です。したがって全角の場合はその半分になります。

図中の文字ボックスは，画面上で 1 つの文字を表現するために使用する四角の範囲です。たとえば，画面が横 80 行，縦 25 行の画面には，80 × 25 の文字ボックスがあるという意味になります。画面上に文字ボックスが存在する場所を文字位置といいます。文字ボックスは画面上の点の数で大きさを表します。

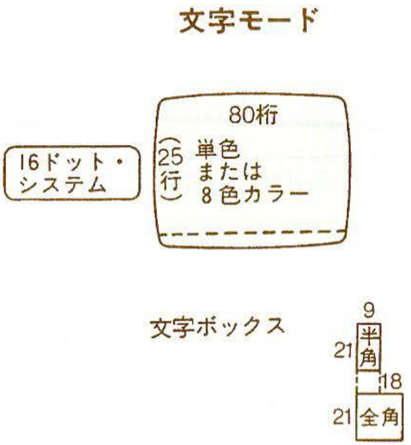
画面の図中点線で表している最後の行は「キーボード・シフト状況およびかな漢字変換域」です。行数はこの最後の行も含めた値です。

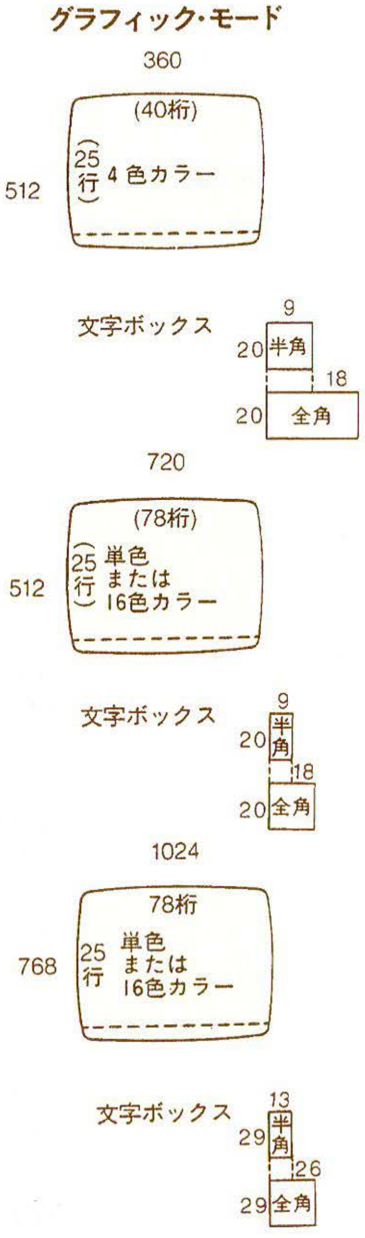
グラフィック・モードの場合は画面の座標と文字位置の両方を表しています。画面の外側に記した値は，その座標系の点の数を表します。() の中の値は文字の桁数または行数です。

単色システム

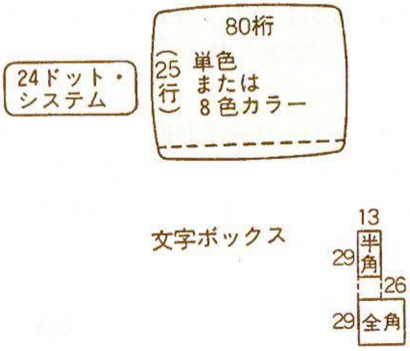

文字モード
グラフィック・モード
16ドット・システム
80桁
25行
720
(80桁)
512
(25行)
文字ボックス
9
21
半角
18
21
全角
文字ボックス
9
20
半角
18
20
全角
24ドット・システム
80桁
25行
1024
768
(25行)
文字ボックス
13
29
半角
26
29
全角
文字ボックス
13
29
半角
26
29
全角

カラー・システム



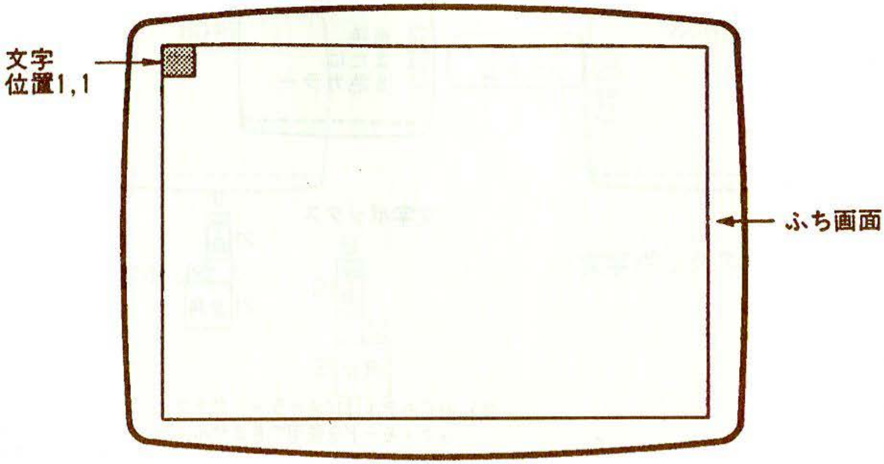


注）旧Cモデルは16色カラー・グラフィック・モードを使用できません。



文字モード

画面を図で表すと次のようです。



文字は，横方向 80 桁表示できます。縦方向は 25 行です。この文字位置には横方向 1 ～ 80，たて方向 1 ～ 25 の番号がついています。この番号は LOCATE ステートメントで文字の位置を示すために使います。また POS (0) や CSRLIN 関数で返る値です。

文字はカーソルの位置に表示されます。23 行目で改行されると，1 ～ 23 の行が，1 行だけ上へスクロール（画面移動）し，行 1 にあった内容は画面から消えます。その結果行 23 はブランクになり，カーソルは行 23 に残っていて表示を続けることができます。

行 24（下から 2 行目）は通常 PF キーの表示用（KEY ステートメントによる）に使われ，この行のスクロールは行われません。ただし，PF キー表示をオフにすれば（KEY OFF で），画面上のこの区域に書き込むことができ，スクロールも行われます。

行 25（最後の行）はキーボード・シフト状況またはかな漢字変換域です。こ

の行はシステムが使用するために予約されています。この行はかな漢字変換のときに使用します。

ディスプレイは文字用に使われる区域の外側に「オーバー・スキャン」区域を持っています。このオーバー・スキャン区域は，ふち画面と呼ばれます。このふち画面を文字表示用に使用することはできません。

次のステートメントは，画面の情報を定義または変更するために使用するものです。

```
CLS       SCREEN
COLOR     WIDTH
LOCATE    WRITE
PRINT     VIEW PRINT
```

グラフィック・モード

グラフィックに使うステートメントには次のものがあります。この内，COLOR と SCREEN ステートメント以外は，グラフィック・モードでのみ使用します（GET と PUT ステートメントはランダム・ファイルのアクセスでも使用しますがグラフィックの場合は全く別の構文です）。

CIRCLE	PAINT	SCREEN
COLOR	PALETTE	VIEW
DRAW	PSET	WINDOW
GET	PRESET	
LINE	PUT	

グラフィック関数は次の 2 つです。

PMAP　　POINT

画面の座標系は，X 方向，Y 方向それぞれ **0～（画面の点の数-1）** で画面の左上隅は座標の（0，0）です。

座標指定

グラフィック・ステートメントを使う場合，画面上のどの点（ドット）に表示するかを指定することが必要です。これは座標の形式で指定します。座標指定には，次の 2 つの形式があります。

絶対形式　（x，y）の形で画面上の実際の座標を指定します。x は水平方向位置，y は垂直方向位置です。

相対形式　STEP（x，y）の形で最終座標からの座標を指定します（「最終座標」は，各グラフィック・ステートメントを使用するたびに設定される最後の点です）。

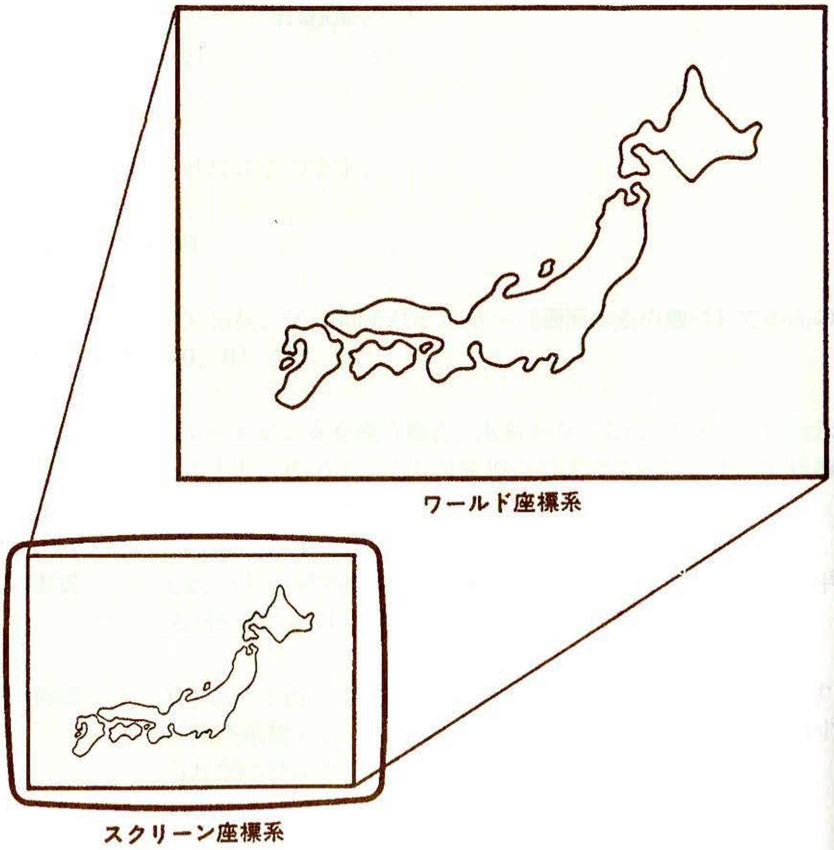
画面の限界を越えて描かないように注意してください。最終座標が混乱する恐れがあります。

次のステートメントは画面上に2つの点を描きます。この点の画面上の座標値は（200, 100）と（210, 80）です。

```
SCREEN 1
PSET (200,100)      '--- 絶対形式
PSET STEP (10,-20) '--- 相対形式
```

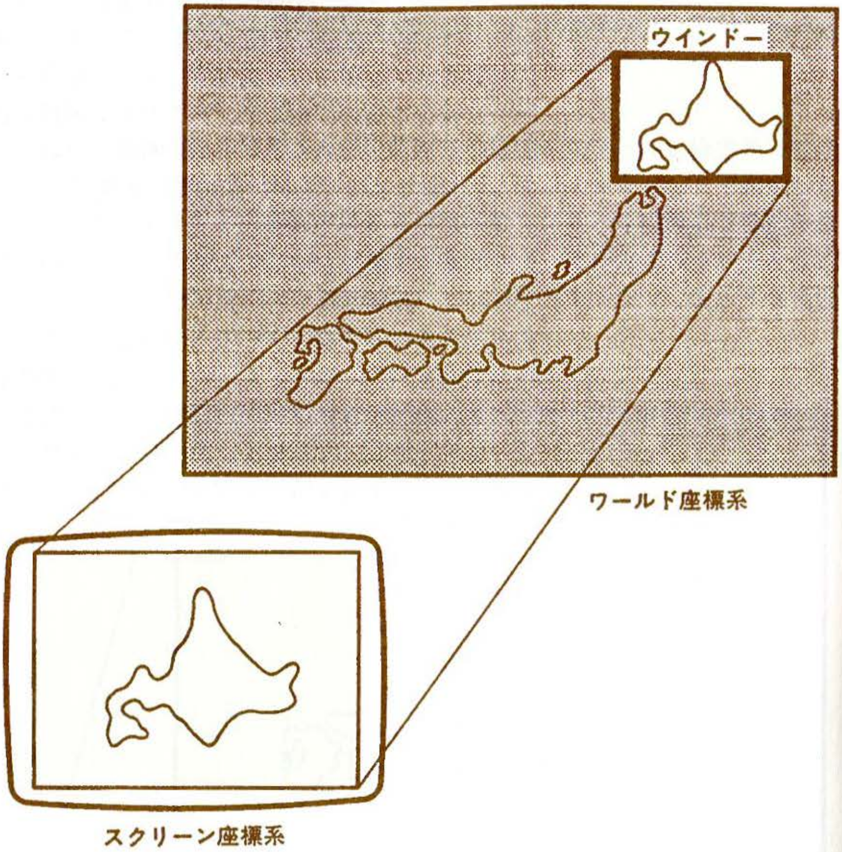
ウインドーとビューポート

グラフィック機能を使用するときに想定する座標系をワールド座標系（論理座標系）といいます。それに対して，SCREEN ステートメントで指定した画面モードによって使用範囲が決まる座標系をスクリーン座標系（物理座標系）といいます。たとえば，今，日本地図を持っていて画面に表示すると仮定します。この場合の日本地図の座標系は，ワールド座標系です。表示する画面の座標系は，スクリーン座標系です。

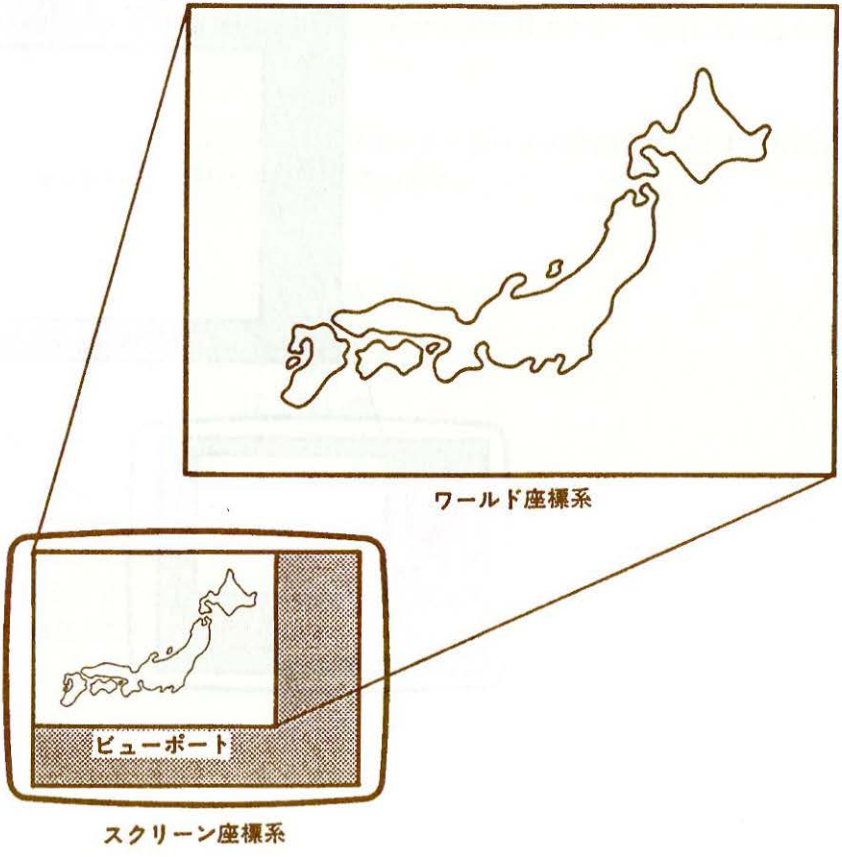


スクリーン座標系の有効範囲は，SCREEN ステートメントで指定した各画面モードが持つ座標系の範囲です。この 2 つの座標系の最大範囲は，ワールド座標系がプログラムで指定する場合の数値が持ちえる値（± 1.7014E + 38）に依存するのに対して，スクリーン座標系は画面モード座標系の範囲に依存します。したがってワールド座標系の場合，グラフィック機能を使用する人が想定するほとんどの座標系は表現できるわけです。

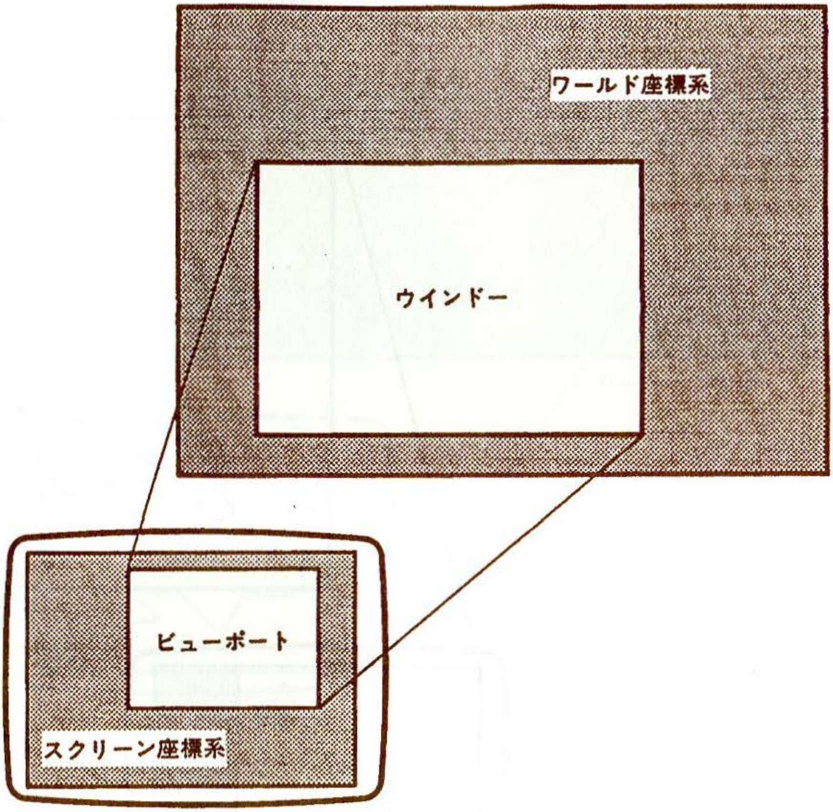
グラフィック機能を使用する場合に，思い描く図形の座標値でそのまま表示できれば，座標変換する必要がなくなります。また想定される座標系すべてを表示するのみでなく，その一部を表示するときもあります。このような場合 WINDOW ステートメントで**ワールド座標系の中の表示範囲**を指定します。この範囲を**ウインドー**といいます。したがってウインドーの最大範囲はワールド座標系になります。次の図は日本地図で北海道を表示する場合の例です。ワールド座標系となる日本地図において北海道のある部分がウインドーです。


ウインドー
ワールド座標系
スクリーン座標系

スクリーン座標系において，画面上の表示範囲を指定することもできます。これは，VIEW ステートメントで行います。この**スクリーン座標系における表示指定範囲をビューポート**といいます。表示領域の最大範囲は，スクリーン座標系です。次の図はスクリーン座標系の左上に日本地図の表示部分を指定した例です。



ワールド座標系，スクリーン座標系，ウインドー，ビューポートの関係をまとめると次の図のようになります。



色

BASICには指定できる色の集合が二通りあります。一つは文字モードと4色カラー・グラフィック・モードのための16色で，もう一つは16色カラー・グラフィック・モードのための64色です。色には色番号が付いていますが，2つの集合の色番号は同じ色でも異なりますから混用しないように注意してください。

文字モードと4色カラー・グラフィック・モードのための16色は，次のように0～15の色番号が付けられています。

0	黒	8	灰色
1	青	9	薄い青
2	緑	10	薄い緑
3	水色	11	薄い水色
4	赤	12	薄い赤
5	紫	13	薄い紫
6	黄	14	薄い黄
7	白	15	明るい白

8～15までの色は0～7までの色を高輝度化したもので色合いは明るく，やや淡色となります。

16 色カラー・グラフィック・モードで使用できる 64 色は，6 つの配合要素で構成されています。配合要素とその重み付けは次のとおりです。

配合要素	暗い赤	暗い緑	暗い青	赤	緑	青
重み	32	16	8	4	2	1

この配合要素で作成された 64 色の色相と色番号は，次のような関係を持っています。図中の数字は色番号です。

0 黒	8 暗い青	16 暗い緑	24 暗い水色	32 暗い赤	40 暗い紫	48 暗い黄	56 灰色
1 青	9 明るい青	17	25	33	41	49	57 薄い青
2 緑	10	18 明るい緑	26	34	42	50	58 薄い緑
3 水色	11	19	27 明るい水色	35	43	51	59 薄い水色
4 赤	12	20	28	36 明るい赤	44	52	60 薄い赤
5 紫	13	21	29	37	45 明るい紫	53	61 薄い紫
6 黄	14	22	30	38	46	54 明るい黄	62 薄い黄
7 白	15	23	31	39	47	55	63 明るい白

たとえば色番号9（明るい青）は，色番号1（青）を配合した色です。このように64色は行方向，列方向それぞれ基本となる1行1列目を配合して作成しています。

参考 次のプログラムを実行すると，実際の色と色番号を確認することができます。

```
100 '------------------
110 '        色
120 '------------------
130 '
140 '--- システムを調べる ---
150 '
160 KEY OFF : LOCATE,,0 : SCREEN 2,1
170 SFLG = 0   '--- SFLG(=0:24,=1:16)
180 SFLG = ABS(POINT(1023,767))
190 '
200 IF SFLG = 0 THEN DX = 13 : DY = 29   '24ドット・システム
210 IF SFLG = 1 THEN DX =  9 : DY = 20   '16ドット・システム
220 '
230 '--- 色を系統別に表示 ---
240 '
250 SCREEN 2,1 : LOCATE,,0
260 C$(1) = "        赤系統の色         "
270 C$(2) = "        紫系統の色         "
280 C$(3) = "        青系統の色         "
290 C$(4) = "        緑系統の色         "
300 C$(5) = "黄色系統の色と無彩色と淡い色"
310 DATA  4,12,20,28,36,44,52,60,37,45,53,61,32,39
320 DATA  5,13,21,29,37,45,53,61,33,41,49,57,40,47
330 DATA  3,11,19,27,35,43,51,59, 1, 9,17,25, 8,24
340 DATA  2,10,18,26,34,42,50,58, 6,14,22,30,16,48
350 DATA 38,46,54,62, 6,14,55, 0, 7,56,63,31,23,15
360 '
```

```
370 R = DX * 5 : RESTORE 310 : C = 15
380 FOR I = 0 TO 1
390   Y = DY*7 + DY*8*I
400   FOR J = 0 TO 6
410     X = DX*7 + DX*11*J
420     CIRCLE (X,Y),R,C
430     READ COL : PAL = (J+1)+(I*7)
440     PALETTE PAL,COL
450     PAINT (X,Y),PAL,C
460 NEXT J,I
470 '
480 '--- 色の切り換えと色番号の表示 ---
490 '
500 POKE 106,0 : K$ = "" : COL = 0 : IDX = 1
510 RESTORE 310
520 '
530 '--- 色の切り換え
540 '
550   LOCATE 3,28
560   PRINT C$(IDX);
570   PAL = 1 : SIGN = 0
580   FOR J = 0 TO 1
590     ROW = 11+J*8
600     FOR K = 0 TO 6
610       READ COL
620       PALETTE PAL,COL
630       LOCATE ROW,7+K*11
640       PRINT USING "##";COL;
650       PAL = PAL + 1
660       COL = COL + 1
670   NEXT K,J
680   LOCATE 24,1  : PRINT "表示している番号は色番号です。";
690   LOCATE 24,59 : PRINT "G:進む B:戻る";
700 '
```

```
710 '--- 進む,戻るのキー入力を待つ
720 '
730   K$ = INKEY$ : IF K$ = "" THEN 730
740   IF K$ = "G" OR K$ = "g" THEN 800
750   IF K$ = "B" OR K$ = "b" THEN SIGN = 1 : GOTO 850
760   GOTO 730
770 '
780 '--- 進む処理
790 '
800   IF IDX = 5 THEN 960
810   IDX = IDX + 1
820 '
830 '--- 戻る処理
840 '
850   IF SIGN = 0 THEN 920
860   IF IDX = 1 THEN 960
870   IF IDX = 2 THEN RESTORE 310
880   IF IDX = 3 THEN RESTORE 320
890   IF IDX = 4 THEN RESTORE 330
900   IF IDX = 5 THEN RESTORE 340
910   IDX = IDX-1
920 GOTO 530
930 '
940 '--- 終了 ---
950 '
960 SCREEN 0,0 : COLOR 0,0,0 : LOCATE,,1
970 END
```

色の使用と画面モード

BASICには，次の画面モードがあります。

- 単色文字モード
- カラー文字モード
- 単色グラフィック・モード
- 4色カラー・グラフィック・モード
- 16色カラー・グラフィック・モード

使用できる画面モードは，ハードウェア構成により異なります。前述の「画面モード」で確認してください。

ここでは各モードの機能を説明します。

単色文字モード

BASICを始動したときと，SCREENステートメントで0，0を指定した場合，単色文字モードになります。単色文字モードでは，下線づけ，高輝度，明滅，反転，罫線の文字属性が指定できます。これらの文字属性はCOLORステートメントで指定します。

次のプログラムは，各文字属性を指定した例です。

```
10 T$ = "単色文字モード"
20 SCREEN 0,0        '--- 単色文字モード
30 COLOR 1           '--- 下線つき文字
40 LOCATE 2,3 : PRINT T$;
50 COLOR 8,1,3       '--- 高輝度，反転，縦横罫線
60 LOCATE 6,10 : PRINT "                ";
70 LOCATE 7,9  : PRINT "                  ";
80 LOCATE 8,8  : PRINT "                    ";
90 COLOR 0,1,0       '--- 反転のみ
100 LOCATE 9,10 : PRINT "                ";
```

```
110 LOCATE 10,10 : PRINT "      ";
115 LOCATE 10,20 : PRINT "       ";
120 LOCATE 11,10 : PRINT "                    ";
130 LOCATE 5,21 : PRINT "  " : LOCATE 6,21 : PRINT "  ";
140 COLOR 16,0,0 '--- 明滅
150 LOCATE 4,21 : PRINT "○";: LOCATE 3,23 : PRINT "○";
160 COLOR 0,0,0  '--- 属性を元に戻す
```

カラー文字モード

SCREENステートメントで0，1を指定するとカラー文字モードになります。カラー文字モードでは，文字の色，罫線の色，罫線を指定できます。指定はCOLORステートメントで行います。

文字の色は色番号0～7までと，8～15までの2組に大別されます。一度に同時に8色までの色を表示することができますが，2組のうち異なる組の色を同時に表示することはできません。色番号0～7と8～15は，対応していて最後に指定したどちらかの組が選択されます。

罫線色は，文字の色とは無関係に，16色の中から任意の1色を選択することができます。

次のプログラムは色を指定した文字を表示します。色番号が0～7の文字を表示したのち，色番号が8～15の文字を表示させるとすでに表示されていた色番号0～7の文字は色番号8～15の文字になります。次のプログラムはこの色の切り換えを繰り返し行っています。

```
10 SCREEN 0,1 : KEY OFF  '--- カラー文字モード
20 C1 = 0 : C2 = 7
30 FOR J = 1 TO 30
40   IF J>16 THEN C1 = 8 : C2 = 15
50   FOR I = C1 TO C2
60   COLOR I,3,3
70   PRINT "BASIC";
80 NEXT I,J
90 GOTO 20
```

このプログラムを終了させるには[Break]キーの機能を使用してください。またカラー文字モードにおいて，COLOR ステートメントで一度変更した文字や罫線の属性を元に戻すには，

```
COLOR 7,0,0
```

を実行してください。

単色グラフィック・モード

SCREEN ステートメントで 1，0 または 2，0 を指定すると単色グラフィック・モードになります。各グラフィック・ステートメントのパラメーター**パレット番号**には，0 または 1 を指定します。0 は表示されないこと，1 は表示することを表します。ただし，単色グラフィック・モードには，パレットがありません。したがって，PALETTE ステートメントは使用できません。単色グラフィック・モードでは，COLOR ステートメントで，下線づけ，反転，罫線の文字属性を指定できます。

次のプログラムは，単色グラフィック・モードでグラフを描いた例です。

```
10 T$ = " *** SIN 関数のグラフ *** "
20 '--- 初期設定 ---
30 SCREEN 2,0        '--- 単色グラフィック・モード
40 KEY OFF : CLS
50 '--- 表題の表示 ---
60 COLOR 0,1         '--- 文字反転の指定
70 LOCATE 2,26 : PRINT T$;
80 COLOR 0,0         '--- 文字属性を元に戻す
90 Y = 348 : N = 1.6           '--- 24ドット・システム用
100 T = POINT (1023,767)  '--- システムを調べる
110 IF T <> -1 THEN 130
120 Y = 240 : N = 1.1          '--- 16ドット・システム用
130 '--- SIN カーブを描く ---
140 LINE (0,Y)-(628*N,Y)
150 LOCATE 14,1 :PRINT "0"  :LOCATE 14,19:PRINT "90"
160 LOCATE 14,38:PRINT "180":LOCATE 14,58:PRINT "270"
```

```
170 LOCATE 14,76:PRINT "360"
180 PSET (0,Y)
190 FOR X = 0 TO 628
200   LINE -(X*N,Y-200*SIN(X/100))
210 NEXT X
```

4色カラー・グラフィック・モード

16ドット・システムでは，SCREENステートメントで1，1を指定すると4色カラー・グラフィック・モードになります。4色カラー・グラフィック・モードでは，0～3のパレット番号を持った4区画のパレットがあります。パレットの4区画には16色の中から任意の1色を指定します。

パレットに色を設定するときや，すでに表示されている色を変更するには，COLORステートメントまたはPALETTEステートメントを使用します。ただし，パレットには省略時の色が設定されているため，COLORやPALETTEを使用しなくてもグラフィック・ステートメントで図形を描くことができます。省略時の色は次のとおりです。

4色カラー・グラフィック・モードの省略時のパレット

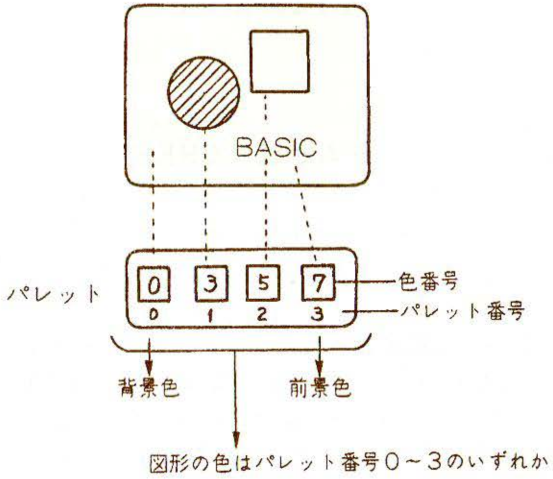
パレット番号	色番号	色
0	0	黒
1	3	水色
2	5	紫
3	7	白

各グラフィック・ステートメントのパラメーター**パレット番号**ではパレットの位置を0～3のパレット番号で指定します。

PALETTEステートメントの使用方法は次の16色カラー・グラフィック・モードの例を参照してください。

背景色は常にパレット番号0の色が使用されます。文字や罫線の色は常にパレット番号3の色が使用されます。

４色カラー・グラフィック・モード



16色カラー・グラフィック・モード

SCREEN ステートメントで 2，1 を指定すると 16 色カラー・グラフィック・モードになります。16 色カラー・グラフィック・モードでは，0 ～ 15 のパレット番号を持った 16 区画のパレットがあります。パレットの 16 区画には 64 色の中から任意の 1 色を指定します。

パレットに色を設定するときや，すでに表示されている色を変更するには，COLOR ステートメントまたは PALETTE ステートメントを使用します。ただし，パレットには省略時の色が設定されているため，COLOR や PALETTE を使用しなくてもグラフィック・ステートメントで図形を描くことができます。

省略時の色は次のとおりです。

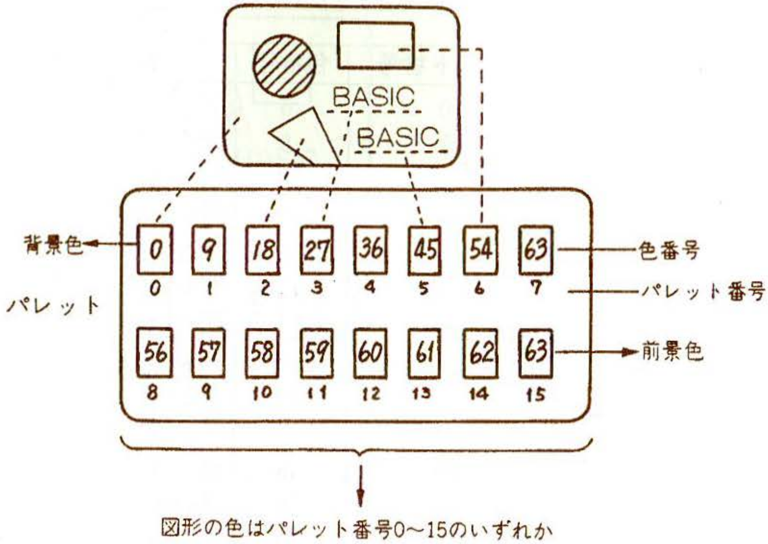
16 色カラー・グラフィック・モードの省略時のパレット

パレット番号	色番号	色
0	0	黒
1	9	明るい青
2	18	明るい緑
3	27	明るい水色
4	36	明るい赤
5	45	明るい紫
6	54	明るい黄
7	63	明るい白
8	56	灰色
9	57	薄い青
10	58	薄い緑
11	59	薄い水色
12	60	薄い赤
13	61	薄い紫
14	62	薄い黄
15	63	明るい白

各グラフィック・ステートメントのパラメーター**パレット番号**ではパレットの位置を 0 ～ 15 のパレット番号で指定します。

背景色は常にパレット番号 0 の色が使用されます。文字や罫線の色は常にパレット番号 15 の色が使用されます。

16色カラー・グラフィック・モード



次のプログラムはPALETTEステートメントでパレットの切り換えを行った例です。

```
10 '*** 16色カラー・グラフィック・モード ***
20 SCREEN 2,1  '--- 16色カラー・グラフィック・モード
30 '--- 暖色系の色 ---
40 DATA 0,4,12,20,28,36,37,38,44,46,52,53,54,60,62,63
50 '--- 寒色系の色 ---
60 DATA 0,3,9,11,13,19,25,27,23,25,43,49,51,57,59,63
70 '--- 初期設定 ---
80 DIM COL%(15,1)
90 FOR I = 0 TO 1
100   FOR J = 0 TO 15
110     READ COL%(J,I)
120   NEXT J
130 NEXT I
140 '--- 箱を描く ---
```

```
150 PALETTE USING COL%(0,0)    '--- 暖色系のパレット
160 FOR N = 0 TO 30
170   LINE -(RND*1023,RND*767),INT(RND*15),BF
180 NEXT N
190 FOR I = 0 TO 3000 : NEXT
200 PALETTE USING COL%(0,1)    '--- 寒色系のパレット
```

画面表示

グラフィック・モードにおいて図形を表すすべての点は，各々色の情報を持っています。この点の『色を表現するために必要なビットを画素ビットといいます。』1つの点の色を表す画素ビットの数は，その画面モードが図形を描くために，同時に最高何色表示できるかに依存しています。したがって，同時に4色表示できる画面モードでは，4色の色を表現するために1つの点につき2ビット必要です。また，同時に16色表示できる画面モードでは4ビット必要です。単色の場合は1ビットで表現できます。

画面モード	画素ビット数
単色	1
4色カラー	2
16色カラー	4

したがって，1つの画面モードを使用するためには，

画面上の点の数 × 画素ビット数

のビットが必要となります。

グラフィック・モードで図形やこれらの色を表現するためにIBM5550システムでは，全点アドレス可能（APA）バッファーを持っています。画面1枚に対するAPAバッファーの使用領域は，システムや画面モードに依存します。単色グラフィックと4色カラー・グラフィックは同じ大きさのAPAバッ

ファーを使用します。4 色カラーは 1 つの点を表示するために 2 ビット必要なので画面の横方向の点の数は単色時の半分になります。

このため，縦方向の座標値と横方向の座標値を等しく指定した場合に X 方向は Y 方向の 2 倍の距離で表示されます。

16 色カラー・グラフィックは旧 C モデル以外のカラー・システムで使用できます。これらのシステムは単色時の 4 倍の APA バッファーを使用しています。このため，16 色カラーは 1 つの点を表示するために 4 ビット必要ですが，単色時と同じ点の数で画面を使用することができます。すなわち，単色および 16 色カラーは，縦横比 1:1 の物理座標で図形を描くことができるグラフィック・モードです。

付録A

ディスク入出力

プログラム・ファイルの管理

プログラム・ファイルを管理するためのコマンドには，以下のものがあります。これらのコマンドは，予約語の後ろに対象となるファイルの**ファイル指定**を記述します。

プログラムの保管

SAVE コマンド

現在記憶域にあるプログラム・ファイルをディスクに保管するためには，SAVE コマンドを使用します。

SAVE コマンドを使用すると，次の 3 つのいずれかの形式でファイルを保管できます。

- 圧縮 2 進形式.........SAVE **ファイル指定**
- 文字形式.............SAVE **ファイル指定**，A
- コード化 2 進形式.....SAVE **ファイル指定**，P

圧縮2進形式は，記述したプログラムの各命令を BASIC がより効率よく解釈できるよう変換して保管されます。普通プログラムはこの形式で保管します。文字形式で保管したファイルより少ない容量で保管することができます。

文字形式は，内部コード形式で保管されます。そのため DOS の TYPE コマンドや EDLIN などの，文字形式ファイルを扱うエディターで記述したそのままの文字を見ることができます。文字形式のファイルを保管する場合**圧縮 2 進形式**のファイルより多いディスク空間を必要とします。MERGE コマンドで重ね合わせるファイルは**文字形式**で保管する必要があります。

コード化 2 進形式はファイルを保護するために使用します。この形式で保管したファイルはプログラム・リストの表示や修正を行うことができなくなります。場合によっては，予備として保護なしのファイルも保管しておくと便利です。

プログラムの読み込み LOADコマンド

ディスクにあるプログラム・ファイルを記憶域に読み込むためには，LOADコマンドを使用します。

LOADコマンドは，次のいずれかの形式でファイルを読み込むことができます。

- LOAD **ファイル指定**
- LOAD **ファイル指定**，R

普通は，Rを指定しない形式でプログラムを記憶域に読み込むために使用します。現在記憶域の中にあるプログラムは消去されすべてのファイルがクローズされた後，ディスクにあるプログラムが読み込まれます。

Rは，プログラムを読み込み，続けて実行する指定です。Rを指定した場合，現在記憶域にあるプログラムは消去されますが，ファイルはオープンされたままの状態です。したがって任意の部分に分けたプログラム・ファイルを読み込み，実行することや同じデータ・ファイルをアクセスすることができます。

プログラムの実行 RUNコマンド

現在記憶域にあるかまたはディスクにあるファイルを実行するためには，RUNコマンドを使用します。

RUNコマンドは，次のいずれかの形式でプログラムを実行させることができます。

- RUN
- RUN **ファイル指定**
- RUN **ファイル指定**，R

何も指定しないと現在記憶域にあるプログラムを実行します。記憶域にプログラムが無い場合は何も動作しません。

ファイル指定を指定するとディスクにあるファイルを読み込んだ後，実行します。**ファイル指定**のみ指定した場合は，現在記憶域にあるプログラムを消去した後ディスクからプログラムを読み込みます。

R を指定した場合，現在記憶域にあるプログラムは消去されますが，ファイルはオープンされたままです。この指定は LOAD コマンドと同様です。

重ね合わせ

MERGE コマンド

デイスクにあるプログラムを現在記憶域にあるプログラムと重ね合わせるためには，MERGE コマンドを使用します。

MERGE コマンドは次の形式で指定します。

- MERGE **ファイル指定**

MERGE コマンドで指定するディスク上のファイルは文字形式ファイルとして保管されている必要があります。このコマンドは現在の記憶域の内容を消去しません。2 つのプログラムの行番号は番号順に並べられます。

現在記憶域にあるプログラム　　　　ディスク上のプログラム（文字形式ファイル）

```
10 SCREEN 0                      100 A=A+B
20 CLS           <==MERGE==      200 PRINT A
30 KEY OFF            □
                      □
                      □
                      □
                      □
```

MERGE 後の記憶域

```
10 SCREEN 0
20 CLS
30 KEY OFF
100 A=A+B
200 PRINT A
```

ファイルの削除

KILL コマンド

ディスク上のファイルを削除するためには KILL コマンドを使用します。

- KILL **ファイル指定**

ファイル名の変更

NAME コマンド

ディスク上のファイル名を変更するためには NAME コマンドを使用します。

- NAME **旧ファイル指定** AS **新ファイル指定**

順次ファイル

BASIC で作成しアクセスできるデータ・ファイルには順次ファイルとランダム・ファイルがあります。

順次ファイルは，データを入力した順番に書き込んでいき，書き込んだ順序で読み取っていく形式のファイルです。

順次ファイルの特徴は次のものがあります。

- 読み取りは，いつもデータの先頭から順次に行うためデータをアクセスする場合の柔軟性と速度が限定されています。
- BASIC からデータを追加する場合は，追加個所からデータの最後までを書き直す必要があります。データの最後に追加するのは比較的簡単にできます。ただし，文字ファイルであれば DOS の EDLIN のようなデータ編集プログラムも併用すると，データの追加や変更を容易に行うことができます。
- プログラムの作成手順が比較的簡単です。
- 可変長のレコードを設定することが可能です。

順次ファイルをアクセスするためには次のステートメントや関数を使用します。

ファイルのオープン

OPEN

データの書き込み

PRINT#（任意の区切り文字を指定する）

PRINT# USING（書式指定が可能）

WRITE#（区切り文字はBASICが入れる）

データの読み取り

INPUT#（項目単位に読み取る）

INPUT$（任意のバイト数で読み取る）

LINE INPUT#（行単位で読み取る）

レコード数

LOC

ファイル長

LOF

ファイルの終わり

EOF

ファイルのクローズ

CLOSE

順次ファイルの書き込み

順次ファイルの書き込み手順を簡単な例と共に説明します。

次のプログラムは入力データ「本の名前」と「作者」の2つの項目を1レコードとした順次ファイルを作成するのためプログラム例です。

```
100 '*** 順次ファイル ***
110 '      (書き込み)
120 '--- ファイルのオープン ---
130 OPEN "DFILE.DAT" FOR OUTPUT AS #1
140 '--- データの入力 ---
150 INPUT "本の名前 ==> ",BN$
160 IF BN$ = "" THEN END
170 INPUT "作者 ==> ",AU$
180 '--- ファイルに書き込む ---
190 WRITE #1,BN$,AU$
200 GOTO 140
```

このプログラムの処理内容は次のようです。

1 ファイル名はDFILE.DAT，出力用，ファイル番号は1でファイルをオープンする。

2 データ（この場合は本の名前と作者）をキーボードから入力する。最初のデータが無いとき（↵キーのみのとき）は処理を終了する。

3 データをファイルに書き込む。

このプログラムを実行して次のデータを入力したとします。

```
RUN
本の名前==>若菜集
作者==>島崎藤村
本の名前==>老人と海
作者==>ヘミングウェイ
本の名前==>
Ok
```

ファイルには次のようにデータが書かれます。

若菜集	島崎藤村	0D	0A	老人と海	ヘミングウェイ	0D	0A	1A

0D：復帰　0A：改行　1A：ファイルの終り（これは制御文字です）

順次ファイルの読み取り

順次ファイルの読み取り手順を例と共に説明します。

次のプログラムは前ページの例で作成した順次ファイルを読み取るためのプログラム例です。

```
100 '*** 順次ファイル ***
110 '    (読み取り)
120 '--- ファイルのオープン ---
130 OPEN "DFILE.DAT" FOR INPUT AS #1
140 '--- データの読みとり ---
150 INPUT #1,BN$,AU$
160 '--- データの表示 ---
170 PRINT BN$;TAB(25);AU$
180 IF EOF(1) THEN END
190 GOTO 140
```

このプログラムの処理内容は次のようです。

1 ファイルをオープンする（ファイル名は DFILE.DAT，入力用，ファイル番号は 1)。

2 データをファイルから読み取る。

3 データを画面に表示する。

このプログラムを実行すると次のように表示されます。

```
若菜集          島崎藤村
老人と海        ヘミングウェイ
```

順次ファイルのプログラム例

順次ファイルを使用した「**売り上げ表**」作成プログラムを紹介します。このプログラムは，商品コードと売り上げデータを入力しファイルを作成するプログラムと，売り上げ表を表示するプログラムの2つからなります。ここでは，商品をハンバーガー・ショップの5品目と限定しています。

プログラムの目的

- 商品コードと売り上げ個数を持つファイルを作成する。
- ファイルを読み，商品ごとの個数と金額を求める。
- 商品コード，商品名，単価，個数，金額，合計個数，合計金額を表示する。

出力例

```
                                        08-26-1986

================================================
                    売り上げ表
================================================
コード      商品名          単価   個数      金額
   1   ハンバーガー        ¥200     3       ¥600
   2   チキンバーガー      ¥280     1       ¥280
   3   フィッシュバーガー  ¥280     3       ¥840
   4   チーズバーガー      ¥250     2       ¥500
   5   ベーコンバーガー    ¥280     1       ¥280
................................................
合計                               10     ¥2,500
```

プログラム・リスト1

```
100 '-----------------------------
110 '    商品コードと個数の入力
120 '-----------------------------
130 '
140 '--- 配列変数の宣言 ---
150 DIM CODE$(99)        '商品コード
160 DIM NUM$ (99)        '個数
170 '--- ファイルのオープン ---
180 OPEN "GOODS.DAT" FOR OUTPUT AS#1
190 '--- データの入力 ---
200 CNT = 1
210 INPUT "商品コード";CODE$(CNT)
220 IF CODE$(CNT) = "" THEN GOTO 200
230 INPUT "個数";NUM$(CNT)
240 CNT = CNT + 1
250 GOTO 210
260 '--- データをファイルに書き込む ---
270 FOR I= 1 TO CNT-1
280   PRINT#1,CODE$(I);" ";NUM$(I)
290 NEXT I
300 END
```

プログラム・リスト2

```
100 '-----------------------------
110 '    集計と売り上げ表の表示
120 '-----------------------------
130 '
140 '--- 配列変数の宣言 ---
150 '
160 OPTION BASE 1
170 DIM GNAME$(5)     '商品名
180 DIM GNSUM (5)     '商品ごとの個数の合計
190 DIM AMOUNT(5)     '商品ごとの金額の合計
200 DIM UNIT  (5)     '単価
210 DIM CODE  (99)    '入力商品コード
220 DIM NUM   (99)    '入力個数
230 SUMN = 0         '個数の合計
240 SUMA = 0         '金額の合計
250 CNT = 1          'データ数
260 '
270 '--- 商品名を配列に読み込む ---
280 '
290 DATA ハンバーガー,チキンバーガー
300 DATA フィッシュバーガー,チーズバーガー
310 DATA ベーコンバーガー
320 FOR I = 1 TO 5
330   READ GNAME$(I)
340 NEXT I
350 '
360 '--- 単価を配列に読み込む ---
370 '
380 DATA 200,280,280,250,280
390 FOR I= 1 TO 5
400   READ UNIT(I)
410 NEXT I
420 '
430 '--- データの入力 ---
440 '
```

```
450 OPEN "GOODS.DAT" FOR INPUT AS #1
460 '
470 INPUT #1,CODE(CNT),NUM(CNT)
480 IF EOF(1) THEN GOTO 510
490 CNT = CNT + 1
500 GOTO 470
510 '
520 '--- 商品ごとに個数を集計する ---
530 '
540 FOR I = 1 TO CNT
550   ID = CODE(I)
560   GNSUM(ID) = GNSUM(ID) + NUM(I)
570 NEXT I
580 '
590 '--- 商品ごとの金額を求める ---
600 '
610 FOR I = 1 TO 5
620   AMOUNT(I) = UNIT(I)*GNSUM(I)
630 NEXT I
640 '
650 '--- 合計を求める ---
660 '
670 FOR I = 1 TO 5
680   SUMN = SUMN + GNSUM(I)   '個数の合計
690   SUMA = SUMA + AMOUNT(I)  '金額の合計
700 NEXT I
710 '
720 '--- データを表示する ---
730 '
740 OPEN "SCRN:" FOR OUTPUT AS #2
750 KEY OFF : SCREEN 0,0 : CLS
760 PRINT#2,SPACE$(35)+DATE$
770 PRINT#2,"========================================"
780 PRINT#2,"              売り上げ表"
790 PRINT#2,"========================================"
```

```
800 PRINT#2,"コード        商品名          単価  個数     金額"
810 FRM$="   #   &                 &¥¥###  #### ¥¥##,###"
820 FOR I = 1 TO 5
830 PRINT#2,USING FRM$;I;GNAME$(I);UNIT(I);GNSUM(I);AMOUNT(I)
840 NEXT I
850 '
860 '--- 合計を表示する ---
870 '
880 PRINT#2,"..............................................."
890 FRM$="合計                            #### ¥¥##,###"
900 PRINT#2,USING FRM$;SUMN;SUMA
910 END
```

ランダム・ファイル

ランダム・ファイルは，ディスク上のファイルをランダムにアクセスできるファイル形式です。

ランダム・ファイルは，**レコード**をアクセスする情報の 1 単位としています。レコードはレコード番号を付けることによって識別できます。1 レコードは最大 32767 バイトの長さまで設定可能です。

ランダム・ファイルの特徴は次のとおりです。

- レコード番号を指定することによってランダムにデータ・ファイルをアクセスできます。
- ランダム・ファイルで扱う数値は，数値を表す 2 進数の内部形式のまま保管することができるため，数値を文字形式で保管するファイルより少ない容量で保管することができます。

ランダム・ファイルに数値を書き込む場合は，ストリングに変換する必要があります。またストリング型の数値を数値として扱う場合，数値に変換する必要があります。

この作業を行うためには，次の関数を使用します。

```
          MKI$
      ===MKS$ ===>
          MKD$
[数 値]                    [ストリング]
           CVI
      <===CVS===
           CVD
```

各関数末尾のI，S，Dはそれぞれ整数，単精度数，倍精度数が対象であることを示しています。MKI＄/MKS＄/MKD＄は数値を2進数の内部形式のままストリングに複写します。したがって，整数値が何桁あってもMKI＄を使用すると2バイトのストリングになります。同様にMKS＄/MKD＄も数値が内部で必要とするバイト数に等しい4バイトと8バイトのストリングを返します。

数値をストリングに変換するためにSTR＄関数を使用することは可能です。ただし，STR＄関数は数値の各桁を1バイトの内部コードに対応させて文字として扱います。したがって，保管する場合のバイト数は数値の桁数に依存し，数値が5桁であれば5バイトのストリングとして保管することになります。この数値が整数値の場合，MKI＄を使用すれば2バイトで保管できるため3バイト少なく保管できます。

MKI＄/MKS＄/MKD＄で数値をストリングに変換した場合は，CVI/CVS/CVDでストリングから数値に変換してください。またSTR＄で変換した場合はVALで変換してください。

一般的に，ランダム・ファイルをアクセスする場合には次のステートメントや関数を使用します。

ファイルのオープン
OPEN

項目の割り振り
FIELD（レコードに対応するバッファー内の各項目を割り振る）

数値をストリングに変換
MKI＄/MKS＄/MKD＄

データをバッファー内に移行
LSET/RSET

レコードの書き込み
PUT

レコードの読み取り
GET

ストリング型の数値を数値型に
CVI/CVS/CVD

レコード番号
LOC（最後に参照されたレコード番号）

ファイルのバイト数
LOF

ファイルのクローズ
CLOSE

ランダム・ファイルの書き込み

ランダム・ファイルの書き込み手順を例と共に説明します。

次のプログラムは1レコードを「商品名」と「金額」の2項目に割り振った，ランダム・ファイル作成の例です。

```
100 '*** ランダム・ファイル ***
110 '        (書き込み)
120 '--- ファイルのオープン ---
130 OPEN "AMT.DAT" AS#1 LEN = 16
140 FIELD #1, 12 AS G$, 4 AS A$
150 '--- データの入力 ---
160 INPUT "レコード番号";REC%
170 IF REC% =< 0 THEN END
180 INPUT "商品名";GOODS$
190 INPUT "金額";AMT
200 '--- バッファーへの移行 ---
210 LSET G$ = GOODS$
220 LSET A$ = MKS$(AMT)
230 '--- ファイルへ書き込む ---
240 PUT #1,REC%
250 GOTO 160
```

このプログラムの処理内容は次のとおりです。

1 ファイルをオープンする（ファイル名は AMT.DAT，ファイル番号は1，レコード長は16バイト)。

2 ランダム・バッファー（1レコードに対応する）内の各項目を割り振る。

3 レコード番号とデータ（商品名と金額）をキーボードから入力する。レコード番号がゼロ以下の場合（[↵]キーのみのとき）は処理を終了する。

4 データをバッファーへ左詰めで移す。数値データ（金額）は，ストリング（この場合は，MKS$ で単精度の数値を 4 バイトのストリングに）に変換する。

5 データをバッファーからファイルに書き込む。

このプログラムは，次のような形のレコードで書き込みをしています。

← 16 バイト →	
商品名（12 バイト）	金額（4 バイト）

プログラムを作成する場合に次の点に注意してください。

- FIELD ステートメントでバッファーの割り振りに使用したストリング変数を，入力ステートメントの中にまたは割り当てステートメントの左辺に置かないようにしてください（LSET や RSET でバッファーに移行する場合以外は）。

  これらのストリング変数のポインターは，FIELD ステートメントによってランダム・ファイル用のバッファー領域を指しています。ところが，上記のように使用するとポインターは通常のストリング変数のための領域を指すようになります。これによってバッファーの割り振りは無効となるため注意が必要です。

ランダム・ファイルの読み取り

ランダム・ファイルの読み取り手順を例とともに説明します。

次のプログラムは前ページの例で作成した順次ファイルを読み取るためのプログラム例です。

```
100 '*** ランダム・ファイル ***
110 '       (読み取り)
120 '--- ファイルのオープン ---
130 OPEN "AMT.DAT" AS#1 LEN = 16
140 FIELD #1, 12 AS G$, 4 AS A$
150 '--- データの読み取り ---
160 INPUT "レコード番号";REC%
170 IF REC% = 0 THEN END
180 GET #1,REC%
190 '--- バッファーからの移行 ---
200 AMT = CVS(A$)
210 '--- データの表示 ---
220 PRINT USING "商品名:&            & 金額:¥¥##,###";G$;AMT
230 GOTO 160
```

このプログラムの処理内容は次のとおりです。

1 ファイルをオープンする（ファイル名は AMT.DAT，ファイル番号は 1，レコード長は 16 バイト)。

2 ランダム・バッファー（1 レコードに対応する）内の各レコードを割り振る。

3 レコード番号を入力する。

4 レコードをファイルからバッファーに読み取る。

5 ストリングに変換した数値は元に戻す（この場合は，MKS $ を使用してストリングに変換しているので CVS を使用)。

6 データを表示する。

プログラム作成するときには次の点に留意してください。

- ファイルからバッファーに読み取ったストリングのデータは，プログラム内で直接使用できます。
- ランダム・ファイルの場合，ファイルへの書き込みと読み取りを同じプログラムの中で行うとき，OPEN ステートメントと FIELD ステートメントは，1 回ずつ指定します。
- ランダム・ファイルの場合，LOC 関数は最後の GET または PUT ステートメントで指定したレコード番号を返します。

ランダム・ファイルのプログラム例

ランダム・ファイルを使用した「**在庫管理**」プログラムを紹介します。このプログラムはメニュー方式で機能を選択します。メニューで選択することによって実行されるプログラムは，すべてサブルーチンになっています。このプログラムの在庫管理は定量発注方式です。

プログラムの目的

- 在庫管理情報を登録する。
- 指定した品目の在庫情報を表示する。
- 在庫を増やす。
- 在庫を減らす。
- 発注点を下回った品目を表示する。

プログラム・リスト

```
100 '--------------------
110 '       在庫管理
120 '--------------------
130 OPEN "在庫.DAT" AS #1 LEN = 29
140 FIELD #1, 1 AS F$, 20 AS D$, 2 AS Q$, 2 AS R$, 4 AS P$
150 CLS : KEY OFF : PRINT : PRINT "機能の選択" : PRINT
160 PRINT "1. ファイルの初期化"
170 PRINT "2. 新しい品目の作成"
180 PRINT "3. 品目ごとの在庫表示"
190 PRINT "4. 在庫を増やす
200 PRINT "5. 在庫を減らす
210 PRINT "6. 発注点を下回った品目の表示"
220 PRINT "7. 終了"
230 PRINT : PRINT
240 INPUT "番号を入力してください ==> ",CC
250 IF (CC<1) OR (CC>7) THEN PRINT "番号が違います":GOTO 150
260 ON CC GOSUB 6000,1000,2000,3000,4000,5000,9999
270 GOTO 150
1000 '
1010 '--- 新しい品目の作成 ---
1020 '
1030 GOSUB 5555
1040 IF ASC(F$) <> 255
   THEN INPUT "品目のデータを書き換えますか(Y/N)";A$:
   IF A$ <> "Y" AND A$ <> "y" THEN RETURN
1060 LSET F$ = CHR$(0)
1070 INPUT "品目";DESC$
1080 LSET D$ = DESC$
1090 INPUT "在庫量";Q%
1100 LSET Q$ = MKI$(Q%)
1110 INPUT "発注点";R%
1120 LSET R$ = MKI$(R%)
1130 INPUT "単価";P
```

```
1140 LSET P$ = MKS$(P)
1150 PUT #1,PART%
1160 RETURN
2000 '
2010 '--- 品目ごとの在庫表示 ---
2020 '
2030 GOSUB 5555
2040 IF ASC(F$)=255 THEN PRINT"該当する品目がありません":RETURN
2050 PRINT USING "品番          ###";PART%
2060 PRINT D$
2070 PRINT USING "現在の在庫量 #####";CVI(Q$)
2080 PRINT USING "発注点       #####";CVI(R$)
2090 PRINT USING "単価      ¥¥##,###";CVS(P$)
2100 FOR I = 1 TO 8000 : NEXT : RETURN
3000 '
3010 '--- 在庫を増やす ---
3020 '
3030 GOSUB 5555
3040 IF ASC(F$)=255 THEN PRINT"該当する品目がありません":RETURN
3050 PRINT D$:INPUT "増やす量";A%
3060 Q% = CVI(Q$)+A%
3070 LSET Q$ = MKI$(Q%)
3080 PUT #1,PART%
3090 RETURN
4000 '
4010 '--- 在庫を減らす ---
4020 '
4030 GOSUB 5555
4040 IF ASC(F$)=255 THEN PRINT"該当する品目がありません":RETURN
4050 PRINT D$:INPUT "減らす量";S%
4060 Q% = CVI(Q$)
4070 IF (Q%-S%) < 0
  THEN PRINT;Q%;"しか在庫が無いため減らせません":GOTO 4050
4080 Q% = Q%-S%
```

```
4090 IF Q% =< CVI(R$) THEN PRINT "現在の在庫量";Q%;
     ", 発注点";CVI(R$):FOR I = 1 TO 8000:NEXT
4100 LSET Q$ = MKI$(Q%)
4110 PUT #1,PART%
4120 RETURN
5000 '
5010 '--- 発注点を下回った品目の表示 ---
5020 '
5030 FOR I = 1 TO 100
5040   GET #1,I
5050   IF ASC(F$) = 255 THEN 5070
5060   IF CVI(Q$)<CVI(R$)
       THEN PRINT D$;"在庫量";CVI(Q$) TAB(40) "発注点";CVI(R$)
5070 NEXT I
5080 FOR I = 1 TO 8000 : NEXT : RETURN
5555 '--- 品番の照合 ---
5560 INPUT "品番(1-100)";PART%
5570 IF PART%<1 OR PART%>100 THEN PRINT "品番の範囲は1-100です"
     : GOTO 5560 ELSE GET #1,PART% : RETURN
6000 '
6010 '--- ファイルの初期化 ---
6020 '
6030 INPUT "初期化を行っても良いですか(Y/N)";B$
6040 IF B$ <> "Y" AND B$ <> "y" THEN RETURN
6050 LSET F$ = CHR$(255)
6060 FOR I = 1 TO 100
6070   PUT #1,I
6080 NEXT I
6090 RETURN
9000 '
9010 '--- 終了 ---
9020 '
9999 CLOSE : END
```

付録B

機械語サブルーチン

機械語サブルーチン処理概略

BASIC インタープリターで機械語サブルーチンを使用するためには次のような処理が必要となります。

- 機械語サブルーチンを記憶域に置く（BLOAD コマンド,POKE ステートメントなどを使用して）。
- 機械語サブルーチンを CALL ステートメントまたは USR 関数で呼び出す。

記憶域に機械語サブルーチンを置く方法は，BASIC のデータ・セグメント内の配列変数の領域を利用する方法を紹介します。

サブルーチンを呼び出す方法には，CALL ステートメントと USR 関数のどちらかを使用しますが，パラメーターの参照手順や複数のパラメーターを受け渡しできるという意味では，CALL ステートメントの方が便利です。

ここでは，アセンブラー・プログラミングの経験者を対象として記述しています。

サブルーチンを記憶域に置く

機械語サブルーチンを呼び出すためには，機械コードを記憶域に読み込んでおく必要があります。機械コードを記憶域に置くときは，DOS や BASIC のデータを壊さないように読み込む必要があります。整数配列の中にサブルーチンを読み込んで行う方法は最も安全なです。

なぜ整数配列なのか

整数配列を使用する利点には，次のものが挙げられます。

- 配列要素に対する割り当てをバイト単位でなくワード単位で行うことができます。ワード単位の割り当て操作は，バイト単位の割り当て操作より容易に行うことができます。また整数変数は，単精度変数や倍精度変数に比べて値の持ち方が複雑でないため，容易に判別できます。

- サブルーチンを配列に割り当てるとき整数配列であれば 2 バイト境界となるためサブルーチンの大きさが 2 の倍数でなくても 1 バイト余るのみです。それに比べて倍精度配列の場合は，最も多くて 7 バイトが無駄になる場合があります。

ここでは，簡単な機械語のコードや機械語サブルーチンのファイルを整数配列に読み込む方法をプログラム例と共に説明します。

サブルーチン・ファイル

小さいサブルーチンは，ファイルとして扱わなくても BASIC のプログラムとして処理することができます。やや長い機械語サブルーチンは，サブルーチンの実行可能なファイルを記憶域に置く必要があります。そのためには，次の方法を用います。

- BLOAD で整数配列にサブルーチンのファイルを読み込みます。BLOAD を使用するためには，BSAVE で保管されたことを示す BSAVE のヘッダーが必要です。この BSAVE のヘッダーを 2 進のファイルに加えて機械語サブルーチンのファイルを読みます。

小さいサブルーチン

比較的小さな機械語のサブルーチンは，BASIC のプログラムの一部として定義することによって簡単に呼び出すことができます。

この場合の機械語のコードは，DATA ステートメントで用意します。コードのデータはサブルーチンとして実行させるために記憶域に置く必要があります。この操作は，次のどちらかの方法で行います。どちらの場合も整数配列を使用します。

- READ ステートメントを使いワード単位で（2 バイトずつ）配列に割り当てる。
- POKE ステートメントを使いバイト単位で（1 バイトずつ）書き込む。

整数配列で注意すべき点は，整数変数としてデータを扱う場合，CPU 内の配列領域において低位バイトと高位バイトが逆転して置かれることです。したがって，ワード単位でアクセスする場合には，1 ワード内のデータを逆に用意する必要があります。POKE で 1 バイトずつ書き込む場合は，単に配列の領域を利用しているのみで BASIC 側から整数変数としてアクセスしないため，コードの順番どおりに書き込みます。次にそれぞれの方法をプログラム例と共に説明しますので，これらの点に注意して参照してください。

READで

READステートメントを使いワード単位で配列に割り当てるには次のように行います。この例は画面を印刷するプログラム例です。実行すると現在画面に表示されているものをプリンターに出力します。

```
100 '*** 配列にREADで機械語を読み込む ***
110 '    (画面印刷を行うサブルーチン)
120 DEFINT A-Z
130 OPTION BASE 1
140 DIM ARRAY(3)
150 '--- 機械語のデータ ---
160 DATA &HCD55  :' 55H.....PUSH BP
170 DATA &H5D05  :' CD05H...INT 5
180 '            :' 5DH.....POP BP
190 DATA &H90CB  :' CBH.....RET FAR
200 '            :' 90H.....NOP
210 '--- 機械語を配列に読み込む ---
220 FOR I = 1 TO 3
230   READ ARRAY(I)
240 NEXT I
250 '--- 配列の機械語を呼び出す ---
260 SUBRT = VARPTR(ARRAY(1))
270 CALL SUBRT
```

このプログラムの処理内容は，次のとおりです。

1. 変数はすべて整数変数とする（DEF INT)。
2. サブルーチンを読み込むために足りる分だけの配列を宣言する(この場合のデータは6バイトなので配列の要素数では3個分確保)。
3. DATAステートメントに機械語のコードを用意する（1項目を2バイトずつにして)。
4. READで配列に割り当てる。

5 配列の先頭のアドレスをVARPTRで得る。

6 サブルーチンを呼び出す。

宣言する配列の要素数は機械コードの長さに依存します（アセンブラーの命令がどのような機械コードを生成するかは，DOSのDEBUGプログラムを使うと分かります）。整数変数は2バイトなので配列要素数は，機械コードの長さ/2になります。ただし，割り切れない場合は1を足します。

このプログラムは整数配列に1ワードずつ割り当てるため，DATAステートメントの機械コードを低位バイトと高位バイト逆に用意しています。

POKEで

READの例と同じ機械コードをPOKEを用いて配列領域に書き込んでいます。

```
100 '*** POKEで機械語を書き込む ***
110 '(画面印刷を行うサブルーチン)
120 DEFINT A-Z
130 OPTION BASE 1
140 P=0 : I=0 : J=0
150 DIM ARRAY(3)
160 '--- 機械語のデータ ---
170 DATA &H55         :' 55H:.....PUSH BP
180 DATA &HCD,&H05 :' CD05H...INT 5
190 DATA &H5D         :' 5DH.....POP BP
200 DATA &HCB         :' CBH.....RET FAR
210 DATA &H90         :' 90H.....NOP
220 '--- 機械語を書き込む ---
230 P = VARPTR(ARRAY(1))
240 FOR I = 0 TO 5
250   READ J
260   POKE(P+I),J
270 NEXT I
280 '--- 配列の機械語を呼び出す ---
290 SUBRT = VARPTR(ARRAY(1))
300 CALL SUBRT
```

このプログラムの処理内容は次のとおりです。

1. 変数はすべて整数変数とする(DEF INT)。

2. サブルーチンを読み込むために足りる分だけの配列を宣言する(この場合のデータは6バイトなので配列の要素数では3個分確保)。

3. DATAステートメントに機械語のコードを用意する(1項目を1バイトずつに区切って)。

4 配列の先頭のアドレスを VARPTR で得る。

5 READ で機械コードを読み取り，POKE で配列の領域に書き込む。

6 配列の先頭のアドレスを VARPTR で得る。

7 サブルーチンを呼び出す。

この場合の配列は，機械語サブルーチンの領域確保のために使用されるのみで，配列要素単位にアクセスするためには使用していません。POKE は確保した領域に 1 バイトずつ順に書き込むために使用しています。

配列の領域をアクセスする場合は，アクセスする直前にそのつど先頭のアドレスを参照してください。これは，BASIC が BASIC のデータ・セグメントを無駄なく使用するために圧縮整理するので，配列のアドレスも常に変更される可能性があるためです。

サブルーチン・ファイル1

ここでは，機械語サブルーチンのファイルを整数配列の領域に BLOAD で読み込む方法を説明します。

機械語サブルーチンのファイルは次の準備が必要です。

1 原始ファイルをアセンブルし，リンクする。

2 DOS のコマンド EXE2BIN を用いて記憶域イメージのファイルにする。

EXE2BIN を使用する場合，原始プログラムにスタック・セグメントがあってはなりません。そのためにリンク時に警告のメッセージが出されても，無視してください。

記憶域イメージのファイルを記憶域に読み込むためには BLOAD コマンドを使用します。BLOAD するためには，アセンブラー・コードに BSAVE されたことを示す次のような 7 バイトの BSAVE のヘッダーが必要です。

```
AAAA     SEGMENT PARA
         DB       0FDH          ;BSAVEの識別子
         DW       0,0
         DW       TRAILER-HEADER   ;モジュールの大きさ
AAAA     ENDS
```

実行可能コードの先頭にこのセグメントがくるように，このヘッダーのセグメント名には，アルファベット順で最も早い英字で始まる名前を付けてください。

また，コード・セグメントの境界合わせは，この BSAVE ヘッダーのすぐ後に記憶域イメージを置くために BYTE の指定をしてください。たとえば，次のように記述します。

```
BASDATSEG SEGMENT BYTE PUBLIC 'CODE'
```

機械語サブルーチンのコードは BASIC のデータ・セグメントの整数配列に読み込みます。アセンブラー・リスティングに示されたオフセットは，サブルーチンの開始点からの相対値であり BASIC のデータ・セグメントの開始点からの値ではありません。

次に機械語サブルーチンを整数配列に読み込む 2 つの方法をプログラム例で説明します。この 2 つの方法は，パラメーターの受け渡しに違いがあります。最初の方法は，CALL ステートメントでパラメーターの記述をせずに，整数配列をアクセスすることによってパラメーターの受け渡しを行うものです。もう一方は，スタックを介してパラメーターの受け渡しを行ない，CALL ステートメントにパラメーターの記述をするものです。

値の受け渡し 1

整数配列の各配列要素の位置とサブルーチン・コードのパラメーター・リスト内における各パラメーターの位置を一致させると配列要素をアクセスすることによってパラメーターを受け渡すことができます。

次のサブルーチンとそれを呼び出す BASIC のサブルーチンは足し算を行ないます。サブルーチンには，2 つのパラメーター（1 番目と 2 番目のパラメーター）を渡し，足した結果（3 番目のパラメーター）を BASIC のプログラムに返します。

<サブルーチン>

```
AAAA     SEGMENT PARA PUBLIC 'CODE'
         DB       0FDH ;BSAVEの識別子
         DW       0,0
         DW       TRAILER-HEADER   ;モジュールの大きさ
AAAA     ENDS

BASDATSEG SEGMENT BYTE PUBLIC 'CODE'
         ASSUME CS:BASDATSEG,DS:BASDATSEG

HEADER   EQU      $          ;記憶域イメージの開始

SUBRT    PROC     FAR
         NOP                 ;BASICの配列とコードの
                             ;位置を一致させるためのNOP
                             ;デバッグを行うときに
                             ;INT 3である0CCHをいれると
                             ;中断点としても使用できる
         CALL     SKIP       ;スタック'BASE'のオフセットを
                             ;プッシュしておく
```

```
BASE:
WORKA    DW      0         ;1番目のパラメーター
                           ;BASICの配列ARY(3)
WORKB    DW      0         ;2番目のパラメーター
                           ;BASICの配列ARY(4)
WORKC    DW      0         ;3番目のパラメーター
                           ;BASICの配列ARY(5)

SKIP:    POP     BX        ;データ領域'BASE'のオフセット
                           ;を取り出す
         MOV     AX,WORKA-BASE [BX]
                           ;1番目のパラメーターを取り出す
         ADD     AX,WORKB-BASE [BX]
                           ;2番目のパラメーターを加える
         MOV     WORKC-BASE [BX] ,AX
                           ;3番目のパラメーターに渡す
         RET
SUBRT    ENDP
TRAILER EQU      $         ;記憶域イメージの終わり
BASDATSEG ENDS
         END
```

<BASIC のプログラム>

```
100 '*** 機械語サブルーチンの呼び出し ***
110 '        (足算のサブルーチン)
115 ' * 配列でパラメーターを受け渡す方法
120 '--- 変数の初期設定 ---
130 OPTION BASE 1
140 DEFINT A-Z
150 SUBRT = 0
160 DIM ARRAY(10)
170 '--- 配列のアドレスを得る ---
180 SUBRT = VARPTR(ARRAY(1))
190 '--- サブルーチンを配列に読み取る ---
200 BLOAD "CUL.BIN",SUBRT
210 '--- 配列にパラメーターを渡す ---
220 ARRAY(3) = 2 : ARRAY(4) = 3
230 '--- サブルーチンの呼び出し ---
240 CALL SUBRT
250 '--- 結果の表示 ---
260 PRINT ARRAY(5)
```

このプログラム例の BASIC の配列と機械語コードは次のような関係に設定しています。

配列要素	機械語コード
1	・
2	・
3	1番目のパラメーター
4	2番目のパラメーター
5	3番目のパラメーター
6	・
7	・
8	・
9	・
10	・

BASIC の各配列要素の位置と機械語コードのパラメーターの定義位置は，一致している必要があります。アセンブラー・プログラムの NOP は配列要素の 3 番目に 1 番目のパラメーターがくるように機械語コードを調整するためのものです。

アセンブラー・プログラムの次の部分は，コード・セグメント内のオフセットを BASIC のデータ・セグメントからのオフセットに変換して処理をしています。これは，宣言した配列のアドレスの移動に対処する処理です。

```
        CALL    SKIP        ;スタック'BASE'のオフセットを
                            ;プッシュしておく
BASE:
WORKA   DW      0           ;1番目のパラメーター
                            ;BASICの配列ARY(3)
WORKB   DW      0           ;2番目のパラメーター
                            ;BASICの配列ARY(4)
WORKC   DW      0           ;3番目のパラメーター
                            ;BASICの配列ARY(5)
```

```
SKIP:   POP     BX              ;データ領域'BASE'のオフセット
                                ;を取り出す
        MOV     AX,WORKA-BASE [BX]
                                ;1番目のパラメーターを取り出す
        ADD     AX,WORKB-BASE [BX]
                                ;2番目のパラメーターを加える
        MOV     WORKC-BASE [BX] ,AX
                                ;3番目のパラメーターに渡す
```

CALL SKIP は，BASIC のデータ・セグメント内の BASE: のオフセットを知るためのものです。この処理は，次のように表現できます。

CALL SKIP

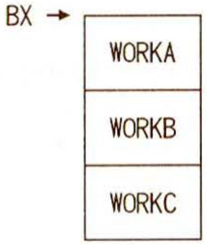
↓

スタック

BASICのDSからのオフセット

↓

POP BX



値の受け渡し 2

スタックを介してパラメーターの受け渡しを行い，CALL ステートメントにパラメーターの記述を行う場合のプログラム例を説明します。

プログラムの機能は，前述と同じものです。

<サブルーチン>

```
PARM      STRUC                   ;パラメーター・リスト
SAVEBP    DW        0             ;BASICのBPレジスター
RETOFF    DW        0             ;リターン・アドレス（オフセット）
RETSEG    DW        0             ;リターン・アドレス（セグメント）
PARMC     DW        0             ;3番目のパラメーターのオフセット
PARMB     DW        0             ;2番目のパラメーターのオフセット
PARMA     DW        0             ;1番目のパラメーターのオフセット
PARM      ENDS

AAAA      SEGMENT PARA
          DB        0FDH          ;BSAVEの識別子
          DW        0,0
          DW        TRAILER-HEADER          ;モジュールの大きさ
AAAA      ENDS

BASDATSEG SEGMENT BYTE PUBLIC 'CODE'
          ASSUME CS:BASDATSEG,DS:BASDATSEG

HEADER    EQU       $             ;記憶域イメージの開始
SUBRT     PROC      FAR
          NOP                     ;デバッグ用
          PUSH      BP            ;BASICのBPレジスターを保管する
          MOV       BP,SP         ;スタックのパラメーターを
                                  ;アドレスできるように

          MOV       SI,[BP].PARMA
                    ;1番目のパラメーターのオフセットを得る
          MOV       AX,[SI]  ;1番目のパラメーターを得る
```

```
        MOV     SI, [BP].PARMB
                ;2番目のパラメーターのオフセットを得る
        ADD     AX, [SI]  ;2番目のパラメーターを加える

        MOV     DI, [BP].PARMC
                ;3番目のパラメーターのオフセットを得る
        MOV     [DI],AX ;結果を返す

        POP     BP          ;BASICのBPを取り出す
        RET     6           ;BASICに戻る
                ;パラメーター3つ分スタックを元に戻す
SUBRT   ENDP
TRAILER EQU     $           ;記憶域イメージの終わり
BASDATSEG ENDS
        END
```

<BASIC のプログラム>

```
100 '*** 機械語サブルーチンの呼び出し ***
110 '          (足算のサブルーチン)
120 ' * CALLでパラメーターを受け渡す方法
130 '--- 変数の初期設定 ---
140 OPTION BASE 1
150 DEFINT A-Z
160 SUBRT = 0 : PARMC = 0
170 DIM ARRAY(12)
180 '--- 配列のアドレスを得る ---
190 SUBRT = VARPTR(ARRAY(1))
200 '--- サブルーチンを配列に読み取る ---
210 BLOAD "CUL1.BIN",SUBRT
220 '--- 配列にパラメーターを渡す ---
230 PARMA = 2 : PARMB = 3
240 '--- サブルーチンの呼び出し ---
250 SUBRT = VARPTR(ARRAY(1))
260 CALL SUBRT(PARMA,PARMB,PARMC)
270 '--- 結果の表示 ---
280 PRINT PARMC
```

前述しましたが，このプログラムの場合も配列の領域にアクセスするときは，アクセスする直前にそのつど配列の先頭アドレスを調べるようにしてください。

サブルーチンの呼び出し

BASIC のプログラムから機械語サブルーチンを呼び出すには，CALL ステートメントまたは USR 関数を使用します。

CALL や USR でサブルーチンを呼び出すときの環境は次のとおりです。

サブルーチンに入るとき，セグメント・レジスター DS と SS は，同じ値で BASIC のデータ・セグメントのアドレス（DEF SEG の省略値）に設定されています。

コード・セグメント・レジスター CS は，最後の DEF SEG で指定された値を持っています。DEF SEG がそれまで指定されていない場合，CS 内の値は，他の 2 つのセグメント・レジスター（DS,SS）内に入っている値と同じ値が設定されています。

引数がストリングの場合，引数に入っているのは，3 バイトのストリング記述子です。ストリング記述子は次の情報を持っています。

0 バイト目	ストリングの長さ（0 ～ 255）
1 バイト目	ストリングのオフセット（下位 8 ビット）
2 バイト目	ストリングのオフセット（上位 8 ビット）

ストリングの実際の情報が入っているのは，ストリング記述子の最後の 2 バイトが示す記憶位置です。

サブルーチンの中でストリング記述子の内容を変更しないでください。

また，サブルーチンの中でストリング自身の内容を変更することはできますが，長さは変更できません。

引数で与えるストリングの内容をサブルーチンの中で変更するときは，BASICのプログラムの内容も変更される場合があるため注意が必要です。次の例を参照してください。

```
A$ = "TEXT"
CALL SUBRT(A$)
```

このように指定していてサブルーチンの中でストリングの内容を変更するとBASICのプログラムの内容も変更してしまう場合があります。

次の例のようにストリングの連結を利用するとBASICのプログラムの内容を変更しないですみます。連結の場合，BASICはストリングをストリング領域内で複写するため，元のストリングとは無関係に変更できます。

```
A$ = "BASIC" + ""
CALL SUBRT(A$)
```

記憶域にサブルーチンを常駐させる方法を用いる場合，サブルーチンから戻るときは，次の点に注意してください。

- BASICはサブルーチンをFAR呼び出しで呼び出します。サブルーチンはFARプロシジャーで作成し，セグメント間RET命令で戻る必要があります。

- 戻るときすべてのセグメント・レジスターとスタック・ポインターは回復されている必要があります。その他のレジスターとフラグは変更されていてもかまいません。

- サブルーチン内で割り込みを停止にした場合は，RETURNで戻る前に割り込み可能状態に戻してください。

CALL ステートメント

CALL ステートメントは次の形式で記述します。

CALL **数値変数** （**変数,・・・**）

数値変数には，DEF SEG で指定したセグメントからのオフセットを指定します。これは，呼び出すサブルーチンの記憶域内の開始点です。

変数はサブルーチンと呼び出すプログラムの間で受け渡す変数の名前です。

CALL ステートメントを指定すると次の動作が行われます。

- かっこ内に記述した各変数に対し，変数の位置がスタックに入れられます。位置は BASIC のデータ・セグメント内のオフセット（2 バイト）です。
- リターン・アドレスとして CS で示されるセグメントとオフセットがスタックに押し込まれます。
- DEF SEG ステートメントで指定されたセグメント・アドレスと，数値変数の値で指定されたオフセットを用いて，機械語サブルーチンに制御が移ります。

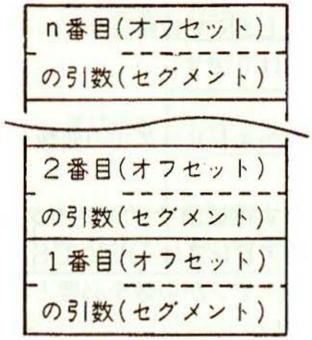
呼び出されるサブルーチンはいくつのパラメーターが渡されるのか知っている必要があります。パラメーターはスタックに積まれます。サブルーチンでパラメーターを参照するためには，現在のスタック・ポインターを BP に移行して参照します。サブルーチンの最初の命令は次のように記述します。

```
PUSH BP      ;BPを保管
MOV  BP,SP   ;BPに移行
```

引数はスタックに次のように積まれます（n 個の引数があるとします）。

CALL SUB（ARG1, ARG2,・・・ARGn）
↓ ↓ ↓
1 番目の引数　2 番目の引数　n 番目の引数

スタック



T をパラメーターの総数とすると，スタック内における n 番目のパラメーターのオフセットは次のようになります。

BP からのオフセット ＝ 2＊(T-n) ＋ 6

CALL ステートメントで呼び出すサブルーチンは，RET n で戻る必要があります。

USR 関数

USR 関数は次の形式で指定します。

USR　n　(引数)

n は DEF　USR で機械語サブルーチンに指定した 0 ～ 9 の番号を指定します。省略値は 0 です。

引数は数式かストリング式の変数名を指定します。

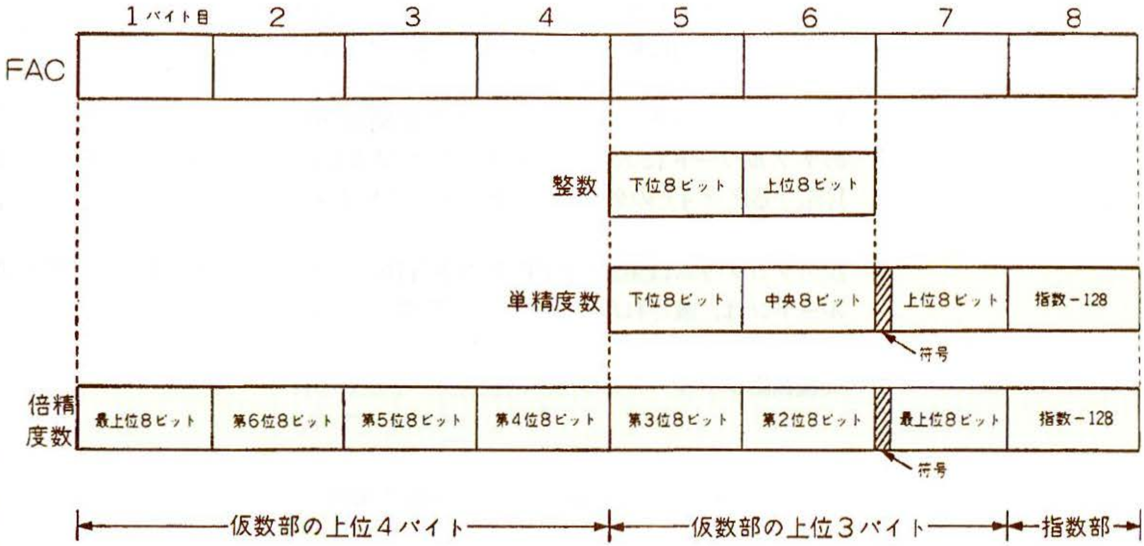
n は呼び出す機械語サブルーチンを指定するために使用します。サブルーチンの開始アドレスは DEF　USR ステートメントでオフセットを指定します。サブルーチンが引数を必要としなくても呼び出しのときには仮引数を与えてください。

USR 関数が呼び出されたときレジスター AL には与えられた引数の型を表す次の値が入っています。

AL 内の値	引数の型
2	2 バイトの整数（2 の補数）
3	ストリング
4	単精度数
8	倍精度数

引数がストリングの場合，DX レジスターは 3 バイトのストリング記述子のアドレスを指しています。

引数が数値の場合，引数の値は浮動小数点アキュムレーター（FAC）内に設定されます。これは BASIC のデータ領域内の 8 バイトの区域です。このとき BX レジスター内の値は 5 バイト目（倍精度数の場合は 1 バイト目）のアドレスを指します。引数は FAC に次のように入ります。



単精度数と倍精度数の場合，FAC の 7 バイト目には先行 1 を抑止し仮数部の最上位 7 ビットが入っています。ビット 7 は引数の符合です（0= 正，1= 負)。

USR 関数は渡された引数と同じ型を返します。返される内容が実数の場合は FAC の内容をそのまま操作しようとすると，操作が繁雑になります。実数値を整数値として扱い，レジスターに置くことができれば操作はより簡単になります。そのために引数を BX に整数として設定する FRCINT，処理した BX の値を整数引数として返す MAKINT という 2 つの処理ルーチンを使用することができます。

MAKINT を使用して返す値を整数にする場合は結果の値を BX レジスターに入れてください。セグメント間リターンの直前で MAKINT を呼ぶと値は FAC 内に置かれ BASIC に返されます。

FRCINT ルーチンのアドレスは USR 関数が呼び出されるとき ES:[BP] のダブルワードに入っています。また MAKINT ルーチンのアドレスは ES:[BP＋4] のダブルワードに入っています。

次のプログラムは FRCINT と MAKINT のアクセス例です。この機械語ルーチンは，渡された引数を 2 倍して返します。

<機械語サブルーチン>

```
AAAA    SEGMENT  PARA
        DB       0FDH       ;BSAVEの識別子
        DW       0,0
        DW       TRAILER-HEADER         ;モジュールの大きさ
        AAAA     ENDS

CSEG    SEGMENT BYTE PUBLIC 'CODE'
        ASSUME CS:CSEG

HEADER  EQU      $          ;記憶域イメージの開始
SUBRT   PROC     FAR
        CALL     DWORD PTR ES:[BP]      ;FRCINTを呼び出す
        ADD      BX,BX      ;値を倍にする
        CALL     DWORD PTR ES:[BP+4]    ;MAKINTを呼び出す
        RET
SUBRT   ENDP
TRAILER EQU      $          ;記憶域イメージの終わり
CSEG    ENDS
        END
```

<BASIC のプログラム>

```
100 '*** USR関数の使用例 ***
110 '(与えられた数値を2倍にする)
120 DEFINT A-Z
130 OPTION BASE 1
140 X = 0 : Y = 0 : Z = 0
150 DIM ARRAY(5)
160 '--- 配列のアドレスを得る ---
170 Z = VARPTR(ARRAY(1))
180 '--- 配列の領域に機械語ルーチンを読み込む ---
190 BLOAD "USR.BIN",Z
200 DEF USR0 = Z
210 X = 5
220 '--- 機械語ルーチンを呼び出す ---
230 Y = USR0(X)
240 PRINT Y
```

機械語サブルーチンで何度も FRCINT と MAKINT を使う場合は，最初に ES と BP を保管しておいてください。

機械語サブルーチンを配列の領域に読み込む場合には，リンクしたあとに DOS のコマンド EXE2BIN を行うことを忘れないでください。

付録C

通信

通信ファイルの入出力

BASIC で RS-232C カードを用いて非同期通信を行う場合は，通信のためのバッファーを介して行います。このバッファーは，ファイルとして扱うことができるため，ディスク・ファイルを扱う場合と同じように処理することができます。

通信ファイルをアクセスするためには，次のステートメントや関数を使用します。入出力ステートメントは，ディスク・ファイルの順次入出力用のステートメントと同じものが使用できます。

ファイルのオープン

OPEN "COM...

データの送信

PRINT#
PRINT# USING
WRITE#

データの受信

INPUT#
LINE INPUT#
INPUT$

バッファー内の文字数

LOC

バッファーの空領域

LOF

バッファーが空か否か

EOF

送信停止と送信開始

非同期通信で難しい点は，入力文字を受信と同じ速度で処理できるようにすることです。1200bps 以上の速度になると「追い付ける」まで相手側のコンピューターからの文字の受信を延期する必要が生じてきます。このときデータを受信している側のコンピューターから，データを送信しているコンピューターへ送信停止要求の合図を送る必要があります。この合図を XOFF といいます。また一時止めていた送信を開始する合図を XON といいます。実際にはこの合図は次の制御文字を送ります。

送信停止（XOFF）　→　CHR $（19）
送信開始（XON）　→　CHR $（17）

この方法は，データの送信を制御するために良く使用される方法ですが，必ずしも他の（本システム以外の）コンピューターにあてはまるものではありません。通信を行う他の相手側のコンピューターとの間で取り交わされている取り決め（プロトコル）に依存しますので注意してください。

入出力関数

通信バッファーの受信状態を把握するためには，前述の入出力関数を使用します。入出力関数の機能の詳細は次のとおりです。

LOC（ファイル番号）	読み取りを待っている入力バッファーの中の文字数を返します。
LOF（ファイル番号）	入力バッファー内の空領域の要領を返します。通信バッファーの大きさから LOC で返る値を引いたものです（通信バッファーの大きさは，BASIC を始動するときに変更できます。省略時は，256 バイトです)。
EOF（ファイル番号）	入力バッファーが空の場合は真（-1）を返し，読み取りを待っている文字がある場合は偽（0）を返します。

Communication buffer overflow（通信バッファーのあふれ）エラーは，入力バッファーがいっぱいになった後に読み取りを行おうとしたときにも起ります。

入力ステートメント

通信ファイルから受信データを読み取る場合，復帰/改行文字などの制御文字も含めた全部のデータを有効とする場合は，INPUT#や LINE INPUT #よりも INPUT $ 関数を使用する方が便利です。INPUT#はカンマや復帰文字を一区切りとします。LINE INPUT#は復帰文字を一区切りとします。

「バッファーに文字があるときは，ある文字すべてを読み取る」という処理を行うには，次のように記述します。

```
100 WHILE NOT EOF(1)          '文字があるときは処理を繰り返す
110 A$ = INPUT$(LOC(1),#1)   'ある文字すべてを読み取る
      .
      .
(A$に返された文字の処理)
      .
      .
500 WEND
```

文字をすばやく処理するために，できるだけ各文字を受信時に検査するのはさけてください。ある特定の文字（制御文字など）を探す場合は，INSTR 関数を使うとストリング中の特定の文字を見付けることができます。

通信端末

データの送受信を行う通信端末のプログラム例を紹介します。このプログラムは，本システムを全２重モードの通信端末として使用したものです。また，このプログラムは，300bps の回線と 256 バイトの通信バッファー（省略時の値）を仮定しています。

```
100 '*** 通信端末 ***
110 '
120 '--- 画面の初期設定 ---
130 SCREEN 0  '文字モード
140 KEY OFF : CLS : CLOSE
150 '--- 値の初期設定 ---
160 DEFINT A-Z 'すべての数値変数を整数型に
170 FALSE = 0              '偽
180 TRUE  = NOT FALSE      '真
190 XOFF$ = CHR$(19)       '送信停止文字の定義
200 XON$  = CHR$(17)       '送信再開文字の定義
210 PAUSE = FALSE          '入力延期のフラグ
220 LFP   = 0              '改行文字の位置
230 '--- 通信ファイルのオープン ---
240 OPEN"COM1:300,E,8"AS #1
250 '--- 画面を出力ファイルとしてオープン ---
260 OPEN"SCRN:"FOR OUTPUT AS #2
270 '--- 通信の処理 ---
280 LOCATE ,,1
290 ON ERROR GOTO 490
300 B$ = INKEY$            'キー入力文字を得る
310 IF B$<>""THEN PRINT #1,B$;
320                        'キー入力文字を送信する
330 IF EOF(1) THEN 300
340    'バッファーに文字が無いときはキー入力を調べる
350 IF LOC(1)>128 THEN PAUSE=TRUE:PRINT #1,XOFF$
360    'バッファーに半分以上入っているときは
370    '受信を延期するためにフラグ(PAUSE)をたてる
380 A$ = INPUT$(LOC(1),#1)  'バッファーの内容を読む
390 LFP = INSTR(LFP+1,A$,CHR$(10))
```

```
400 IF LFP>0 THEN MID$(A$,LFP,1)=" ":GOTO 390
410     '画面に表示するときに行間が空かないよう
420     '改行文字を取り除く
430 PRINT #2,A$;:IF LOC(1)>0 THEN 350
440     '受信文字を表示し,さらにバッファーを調べる
450 IF PAUSE THEN PAUSE=FALSE:PRINT #1,XON$;
460     'XOFFで送信が止められているときは,
470     'XONを送り再開する
480 GOTO 300   'キー入力を調べる
490 PRINT"ERROR NO";ERR:RESUME
500 END
```

すべての変数を整数変数として定義しているのは，可能な限り処理速度を速めたいためです。特にループ変数に整数変数を用いると速度を速めることができます。

未処理の文字が無い場合，行 300 の INKEY $ は空ストリングを返します。

ファイルの送受信

次のプログラムは文字形式ファイルの送受信を行う簡単なプログラム例です。

```
100 '*** ファイルの送受信 ***
110 '
120 '--- 画面の初期設定 ---
130 SCREEN 0                '文字モード
140 KEY OFF : CLS : CLOSE
150 '--- 値の初期設定 ---
160 DEFINT A-Z              'すべての数値変数を整数型に
170 XOFF$ = CHR$(19)        '送信停止文字の定義
180 XON$  = CHR$(17)        '送信再開文字の定義
190 '--- 通信ファイルのオープン ---
200 OPEN"COM1:300,E,8"AS #1
210 '--- 通信の処理 ---
220 LOCATE ,,1
230 ON ERROR GOTO 620
240 '--- ファイル名の送信 ---
250 INPUT "1:受信 2:送信";ANS$
260 IF ANS$ = "2" THEN 320
270 INPUT "ファイル名は(英大文字/日本語)";FIL$
280 IF FIL$<>"" THEN PRINT #1,FIL$
290                         'ファイル名を送信する
300 PRINT #1,XOFF$
310 GOTO 490
320 '--- ファイル名の受信 ---
330 PRINT "*ファイル名の受信*"
340 LINE INPUT #1,FIL$
350 PRINT "要求ファイルは ";FIL$
360 IF FIL$ = "" THEN 340
370 FOR I = 1 TO 8000 : NEXT
380 '--- ファイルを送信する ---
390 PRINT "*ファイルの送信*"
400 OPEN FIL$ FOR INPUT AS #2
410 PRINT "*送信中*"
420 LINE INPUT #2,FLINE$
```

```
430 PRINT #1,FLINE$
440 PRINT FLINE$
450 IF EOF(2)  THEN 470
460 GOTO 420
470 PRINT "*送信終了*" :PRINT #1,"*送信終了*"
480 GOTO 610
490 '--- ファイルの受信 ---
500 PRINT "*ファイルの受信*"
510 OPEN "&&COM.DAT" FOR OUTPUT AS #3
520 PRINT "*受信中*"
530 PRINT #1,XON$
540 LINE INPUT #1,FLINE$    'バッファーの内容を読む
550 PRINT #1,XOFF$
560 IF FLINE$ = "*送信終了*" THEN 600
570 PRINT#3,FLINE$
580 PRINT FLINE$
590 GOTO 530
600 PRINT "*受信終了*"
610 END
620 PRINT"ERROR NO";ERR:RESUME
```

受信したファイルは，&&COM.DAT というファイル名で保管されます。

このプログラムは，送信要求されたファイルを行単位で受信側に送信しています。

プログラミングの注意

半2重と全2重

モデムを介する場合は，通信回線が半2重モード（2線式）であるか全2重モード（4線式）であるかによって通信制御プログラムが異なってきます。非同期通信の場合，半2重モードではモデム状況レジスター（&H3FE）の7ビット目，すなわちDCD（受信キャリア検出）信号は，相手側のRTS（送信要求）信号によりオン/オフされます。したがって，データを送信する場合は自分のモデム状況レジスターのRTS信号がオフになっていることを確認してから，データを送信します。データを送信し終わったら自分のRTS信号をオンにして，相手にもデータを送信する機会を与える必要があります（通常プロトコルを決めて送受信します）。

BASICのPRINT#ステートメントは命令終了後（送信終了後）RTSをオンにしないため，プログラムで制御する必要があります。これは，モデム制御レジスター（&H3FC）をアクセスして行うかまたはINPUT系の命令を実行します（RTS信号は，OPEN "COM...ステートメントの実行時には，ONにしておきます。ただし，プロトコル上でどちらかの端末またはホストが初期状態で必ず送信可能である場合は，この限りではありません）。

通常全2重モードの場合は，半2重モードのようにDCD信号に注意する必要はありません。

交換回線と専用回線

専用回線では，モデムの状態の制御時間を考慮する必要はありませんが，交換回線では，ダイヤルして電話する際，発信音が聞こえるまで時間がかかるように，制御の時間に気を配る必要があります。時間の設定は，OPEN "COM...ステートメントのDS,CSで設定できます。交換回線の場合は，DSを最大65秒まで設定可能ですが，タイム・アウトが起こる場合は，OPEN "COMステートメントを再度実行する必要があります。

かな/漢字の通信

漢字，ひらがな，カタカナのコードを送受信する場合は，8データ・ビット，偶数パリティーにしてください。

雑音対策

通信手段が確立するまで（ダイヤルを回して相手が出てくるまで）の間は，様々な雑音が回線上にのる場合があります。この雑音でプログラムが誤動作しないよう，次のどちらかの方法をとってください。

- 通信手順が確立するまで受信モードに入らず，データを受け取らないようにする。
- 雑音を本当のデータとして扱わないよう，エラー回復ルーチンをプログラム中に設ける。

たとえば，音響カプラーを用いる場合，BASIC のプログラムで OPEN " COM...ステートメントの前に保留を入れておき，保留の間に電話でホストを呼び出し，音響カプラーを設定します。この状態で何らかのキー・インをきっかけとして，プログラムが動き出すようにしておけば通信開始時の雑音を防ぐことができます。

制御信号の操作

本システムで送受信される制御信号は，EIA RS-232C 標準に従って決められたものです。他のコンピューターや周辺装置とのデータ交換を制御するためには，この標準に沿って行う必要があります。信号はオン（+ 3 ボルト以上）またはオフ（-3 ボルト以下）の DC（装置制御）電圧です。ここでは，制御信号の設定状態および操作について説明します。

ただし，多くの場合 BASIC で非同期通信を行うために制御信号を操作しなくても，簡単なデータの送受信を行うことができます。以下の記述は，参考として参照してください。

OPEN 時の出力信号

本システムで BASIC を開始するとき RTS（送信要求）と DTR（データ端末レディ）信号は，オフになっています。これらの信号は，省略時のパラメーターで OPEN "COM...を実行するとオンになります。RTS 信号をオフの状態する場合は，OPEN "COM の RS を指定します。これらの信号がオンの場合は，通信ファイルがクローズされる（CLOSE，END，NEW，RESET，SYSTEM，R なしの RUN によって）までオンのままです。かりに，OPEN "COM...が失敗しても DTR 信号（適用できる場合は RTS 信号も）はオンのままです。したがって，CLOSE を実行しなくても再び OPEN "COM...を試みることができます。

入力制御信号

通常，CTS（送信可）と DSR（データ・セット・レディー）信号のどちらかがオフになっていると OPEN "COM ステートメントは実行されません。1 秒後 BASIC は Device Timeout（装置タイムアウト）エラー（エラー番号 24）を返します。このとき DCD（受信キャリア検出）信号がオン/オフどちらの状態であってもプログラムの操作には，影響がありません。

これらの信号のテストは，OPEN "COM の RS，CS，DS，CD で指定します。

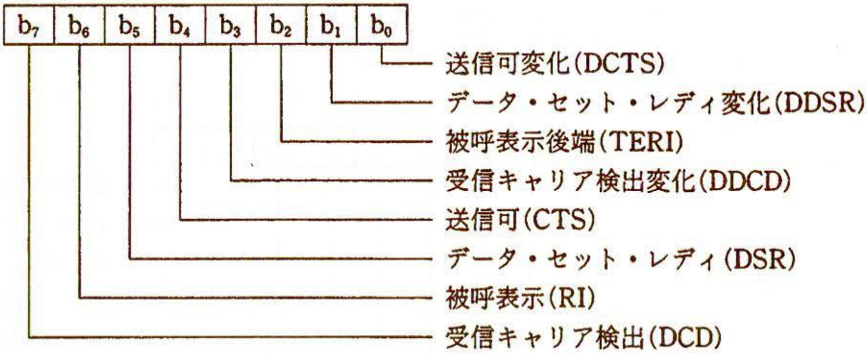
プログラムの実行中にテストされている信号のいずれかがオフになると，通信ファイルに関連づけられた入出力ステートメントは実行されません。たとえば，CTS と DSR 信号がオフになった後に PRINT#ステートメントを実行すると Devices Fault（装置の故障，エラー番号 25）か Device Timeout（装置タイムアウト，エラー番号 24）エラーになります。ただし，このようなエラーが起きても RTS と DTR 信号はオンのままです。

信号のテストをするためには，INP 関数を使用して，RS-232C カードのモデム状況レジスター内にあるビットを読み取ります。

モデム装置制御信号

モデム装置制御信号は，モデム状況レジスターの値を読むことによって調べることができます。

モデム状況レジスター（&H3FE）は，次のとおりです。



モデム状況レジスターの上位 4 ビットは入力信号線の現在の状態を示しています。下位 4 ビットは入力信号線の状態の変化の有無を示しています。

たとえば，受信キャリア検出を調べるには，次のように行います。

```
100 DCD% = INP(&H3FE)
200 DCD% = DCD% AND &H80
```

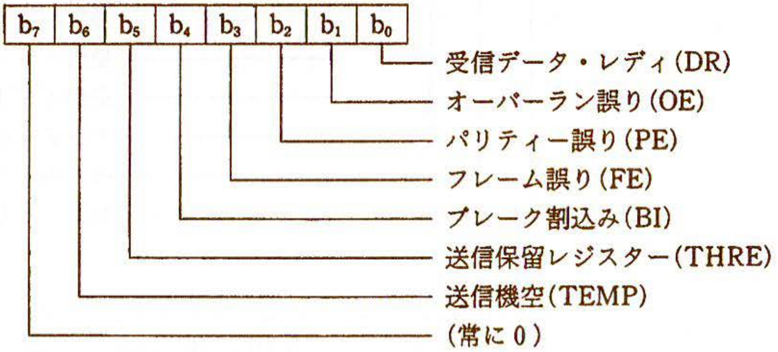
半 2 重モードでは，DCD 信号がオフのときにのみデータを送信しなければならない場合があります。その場合は，DCD%が&H80 であればデータの送信を見合わせ，&H00 ならば開始するようにします。

入力制御信号 CTS，DSR，RI，DCD の内 CTS，DSR，DCD のテストは，OPEN "COM...ステートメントで指定できます。RI は，BASIC の通信では，使用しません。

プログラム内でこれらの信号をテストする必要がある場合このレジスターの各ビットを調べます。このレジスターを読み取るには INP 関数を使用し，INP（&H3FE）と指定してください。

上位 4 ビットの各信号が変化すると，下位 4 ビットの変化を示す信号がオンになります。

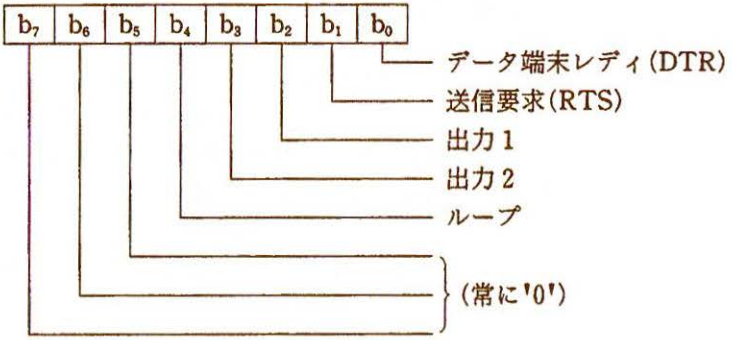
回線の状態を調べるためには，回線状況レジスターを参照します。回線状況レジスター（&H3FD）は，次のとおりです。



このレジスターの値を読み取るには INP（&H3FD）を指定してください。このレジスターを調べることにより通信回線から文字を受け取るときに，どんなタイプのエラーが起こったかを知ることができます。

出力制御信号

RTS（送信要求）や DTR（データ端末レディ）信号を制御するためには，モデム制御レジスターを制御します。モデム制御レジスター（&H3FC）は次のとおりです。



このレジスターに値を書き込むには OUT ステートメントにアドレス&H3FC を指定してください。このレジスター内のビットの多くは OPEN "COM ステートメントが実行されたときに BASIC によって設定されます。

レジスターの値の変更は，通信障害を引き起こすこともあるため，ビットの書き換えには注意が必要です。レジスターの値を書き換える場合には，書き込む前に INP 関数で現在のレジスターの状態を読み取った後，書き換えるビットのみを変更した後書き込んでください。

このレジスターでは，DTR と RTS 信号以外は書き換えないでください。

RTS 信号をオフにするには，次のように行います。

```
100 MCR% = INP(&H3FC)
110 MCR% = MCR% AND &HFD
120 OUT &H3FC,MCR%
```

回線制御レジスター（&H3FB）のビット 6 には中断設定のビットがあります。このビットは，中断信号の送信を制御するビットです。中断は遠隔コンピューターに送信を停止するよう信号を送るときに使用します。普通中断は，0.5 秒続きます。このような信号を作るためには，中断設定ビットをオンにし，中断信号に必要な時間だけ待ち，その後ビットをオフにします。これを BASIC で記述すると次のようになります。

```
100 LCR% = INP(&H3FB)     '回線制御レジスターを読む
110 B% = LCR% OR &H40     '中断設定ビットを読む
120 ' 中断設定ビットがオフの場合はオンとして読む
130 OUT &H3FB,B%          '中断設定ビットをオンにする
140 FOR I = 1 TO 500 : NEXT I
150 OUT &H3FB,LCR%        '回線制御レジスターを最初の状態に戻す
```

通信エラー

通信のエラーは，次の状態のときに起こります。

ファイルのオープン時

- テストされる信号（CTS，DSR，DCD）のいずれかがオフ状態であると，Device Timeout（装置タイムアウト）エラーになります。

データの読み取り時

- オーバーランが起こると Communication overflow（通信バッファーのあふれ）エラーとなります。
- オーバーラン，中断，パリティー，フレーム指示エラーが起こると Device I/O Error（装置入出力エラー）になります。
- DSR または DCD がオフ状態であると Device fault（装置の故障）エラーになります。

データの書き込み時

- BASIC が他の作業をしている間，モデム状況割り込みの発生で CTS，DSR，DCD がオフ状態になった場合 Device Fault（装置の故障）エラーになります。
- データをバッファーに書く作業を待っている間に CTS，DSR，DCD がオフ状態になった場合 Device Timeout（装置タイムアウト）エラーになります。

付録D

コード体系

コード体系

文字にはすべてコードが割り振られています。BASICでは，連続コードと呼んでいる10進数の連続番号とこの連続コードを16進数にして表した内部コードを主として扱います。

文字コードは，1バイト・コード文字セット（連続コード255以下）と2バイト・コード文字セット（連続コード256以上）の2つに別けて説明します。

1バイト・コード文字セットは，JISCII（JIS Code Information Interchange）に相当するものです。2バイト・コード文字セットは，JIS情報交換用漢字符合系に準じています。

1バイト・コード文字は，連続コードと内部コードで識別できます。2バイト・コード文字は，連続コード，内部コード，JIS区点コード，JIS16進コードで識別できます。

連続コード（10進数）

本システムで使用するすべての文字に付けられている10進数の連続番号です。値の有効範囲は，0～11535です。

内部コード（16進数）

本システム内部で文字データとして認識するための16進数のコードです。1バイト・コード文字の場合，内部コードは，連続コードを1バイト（2桁）の16進数で表したものです。2バイト・コード文字の場合は，連続コードを変換規則に従って2バイト（4桁）の16進数に変換したものです。

JIS区点コード（10進数）

JIS 情報交換用漢字符号系の区点番号は，1 つの文字を表すためにその区点番号（1 ～ 94）をハイフンでつないだものです。本システムで使用する JIS 区点コードは，区番号と点番号を各々 2 桁（01 ～ 94）で表し，ハイフンを用いずにつないで 4 桁にしたものです。たとえば，区点番号 1-60 は，本システムの区点コードでは，0160 に相当します。BASIC の中で用いる場合には，コードの先頭に&K をつけます。

JIS16進コード（16進数）

JIS 情報交換用漢字符号系で定められている第 1 バイトと第 2 バイトを 21 から 7E の各々 2 桁の 16 進数で表したもので，各文字は，4 桁の 16 進数で表されます。BASIC の中で用いる場合には，コードの先頭に&J をつけて識別します。

1バイト・コード文字セット

1バイト・コード文字は，連続コード（10進数）0～255で表される文字です。この文字集合は，JISCIIに相当しますが，一部異なるところがあります。また，JISCIIにおける未定義領域のうち81～9F,E0～FCは，2バイト・コードの第1バイトとして使用されます。

1バイト・コード文字には，画面やプリンターに出力するための文字と制御文字があります。制御文字は，データの流れを制御する文字で普通は画面やプリンターなどにデータを出力する場合に使用します。プリンターの制御文字の働きに関する詳細は，各プリンターの解説書を参照してください。BASICから画面に制御文字を送る場合は，PRINTステートメントを，プリンターに制御文字を送る場合は，LPRINTステートメントを使用します。

次ページの表は，1バイト・コード文字セットです。1バイト・コード文字セットについては，次の点に注意してください。

- 連続コードの129～159,224～252（内部コードの81～9F,E0～FC）は，2バイト・コード文字の第1バイト目用に予約されています。この部分の1バイトのみでは，文字は表現されません。2バイト・コード文字の第1バイト目は，必ず内部コード81～9F,E0～FCの範囲にあります。

- 連続コード1～6，16，21～23，25の文字を表示する場合は，PRINT CHR$（連続コード）の形式を使用してください。[前面]キーとテン・キーで入力することは，できません。

- 連続コード29～31の文字は，プリンター出力することはできますが，画面に出力することはできません。これは，これらの文字が画面に対しては制御文字として働くように決められているためです。

連続コード	文　字	連続コード	文　字	連続コード	文　字	連続コード	文　字
000	〈空白文字〉	030	→	060	<	090	Z
001	╔	031	←	061	=	091	[
002	╗	032	〈スペース〉	062	>	092	¥
003	╚	033	!	063	?	093	]
004	╝	034	"	064	@	094	∧
005	║	035	#	065	A	095	_
006	═	036	$	066	B	096	`
007		037	%	067	C	097	a
008		038	&	068	D	098	b
009		039	'	069	E	009	c
010		040	(	070	F	100	d
011		041	)	071	G	101	e
012		042	*	072	H	102	f
013		043	+	073	I	103	g
014		044	,	074	J	104	h
015		045	-	075	K	105	i
016	╬	046	.	076	L	106	j
017		047	/	077	M	107	k
018		048	0	078	N	108	l
019		049	1	079	O	119	m
020		050	2	080	P	110	n
021	╩	051	3	081	Q	111	o
022	╦	052	4	082	R	112	p
023	╣	053	5	083	S	113	q
024	╠	054	6	084	T	114	r
025		055	7	085	U	115	s
026		056	8	086	V	116	t
027	↲	057	9	087	W	117	u
028	↑	058	:	088	X	118	v
029	|	059	;	089	Y	119	w

連続コード	文　字	連続コード	文　字	連続コード	文　字	連続コード	文　字
120	x	160	〈スペース〉	190	セ	220	ワ
121	y	161	。	191	ソ	221	ン
122	z	162	「	192	タ	222	゛
123	{	163	」	193	チ	223	゜
124	|	164	、	194	ツ	224	
125	}	165	・	195	テ	·	
126	―	166	ヲ	196	ト	·	
127		167	ァ	197	ナ	·	
128	〈空白文字〉	168	ィ	198	ニ	·	
129		169	ゥ	199	ヌ	·	
·		170	ェ	200	ネ	·	
·		171	ォ	201	ノ	·	
·		172	ャ	202	ハ	·	
·		173	ュ	203	ヒ	·	
·	2バイト・コード文字の第1バイト	174	ョ	204	フ	·	2バイト・コード文字の第1バイト
·		175	ッ	205	ヘ	·	
·		176	ー	206	ホ	·	
·		177	ア	207	マ	·	
·		178	イ	208	ミ	·	
·		179	ウ	209	ム	·	
·		180	エ	210	メ	·	
·		181	オ	211	モ	·	
·		182	カ	212	ヤ	·	
·		183	キ	213	ユ	·	
·		184	ク	214	ヨ	·	
·		185	ケ	215	ラ	·	
·		186	コ	216	リ	252	
·		187	サ	217	ル	253	
·		188	シ	218	レ	254	
159		189	ス	219	ロ	255	

制御文字と連続コード

連続コードの 0 〜 31 と 127 は，制御文字としても働きます。BASIC から制御文字を送る場合，画面には PRINT ステートメント，ディスク・ファイルには PRINT#ステートメント，プリンターには LPRINT ステートメントを使用します。次の例は制御文字の代表的な使用例です。

「ビープ音を鳴らす」

BASIC でビープ音を鳴らすときは，BEEP ステートメントを使用しますが，これは次のように記述することもできます。

```
PRINT CHR$(7)
```

「プリンターに書式送りを送る」

書式送り（FF）は，最初に設定した，用紙の先頭まで 1 枚分用紙を送ります。これは，単票の場合用紙の排出を意味します。

```
LPRINT CHR$(12);
```

「改行を画面に送る」

画面に改行文字を送る場合は次のように記述します。

```
PRINT "ABC"+CHR$(10)
PRINT "DEF"
```

この結果は次のようになります。

```
ABC

DEF
```

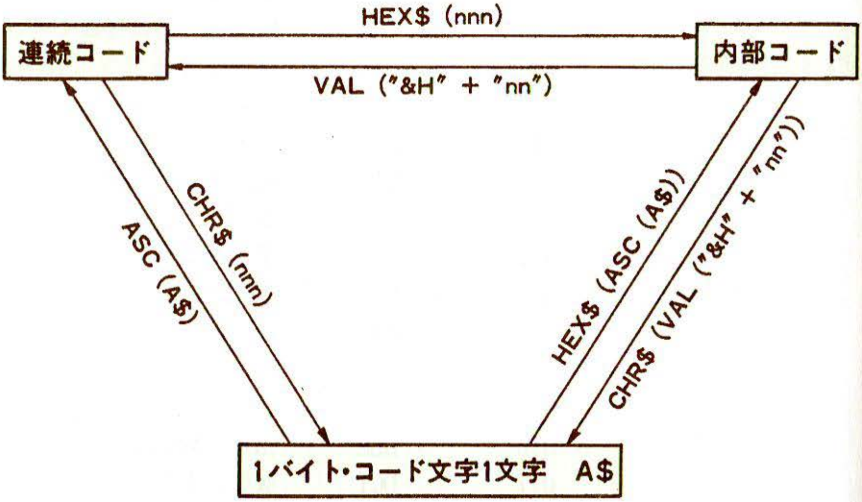
「ファイルの終わりを調べる」

ファイルの終わり（EOF）を示す制御文字は，連続コードの 26 です。文字形式ファイルでファイルの終りを探す場合は，CHR＄（26）を調べます。

連続コード	制御文字	名称
000	NUL	空白
001	SOH	ヘッディング開始
002	STX	テキスト開始
003	ETX	テキスト終了
004	EOT	伝送終了
005	ENQ	問合せ
006	ACK	肯定応答
007	BEL	ベル
008	BS	後退
009	HT	水平タブ
010	LF	改行
011	VT	垂直タブ
012	FF	書式送り
013	CR	復帰
014	SO	シフトアウト
015	SI	シフトイン
016	DLE	伝送制御拡張
017	DC1	装置制御 1
018	DC2	装置制御 2
019	DC3	装置制御 3
020	DC4	装置制御 4
021	NAK	否定応答
022	SYN	同期信号
023	ETB	伝送ブロック終結
024	CAN	取消
025	EM	媒体終端
026	SUB	置換キャラクタ
027	ESC	拡張
028	FS	ファイル分離キャラクタ
029	GS	グループ分離キャラクタ
030	RS	レコード分離キャラクタ
031	US	ユニット分離キャラクタ
127	DEL	抹消

コード間の変換

1バイト・コード文字（1文字），連続コード，内部コードは次の関数を用いて変換することができます。A＄は1文字の1バイト・コード文字です。



2バイト・コード文字セット

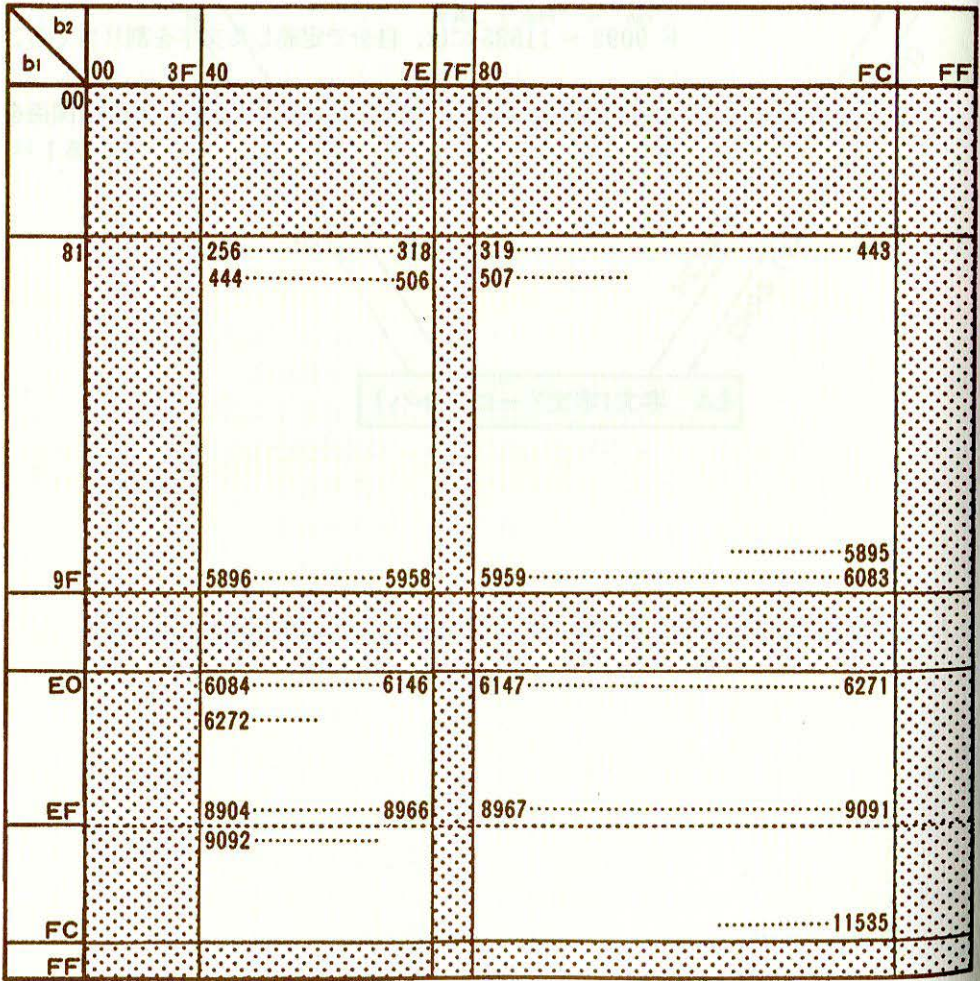
2バイト・コード文字は，連続コード（10進数）256～11535が割り振られている文字の集合です。それぞれの文字は，2バイトの内部コードを持っています。この2バイト・コード文字セットのうち，連続コード256～9091の文字は，JIS情報交換用漢字符号系に対応しているため，この範囲の各文字にはJIS区点コードとJIS16進コードが割り当てられています。連続コード9092～11535には，自分で定義した文字を割り当てることができます。

ここでは，2バイト・コード文字の4つのコード間の関係を内部コードを基本にして説明します。以下の図でb1は内部コードの第1バイト，b2は内部コードの第2バイトを表しています。

連続コードと内部コードの関係

2バイト・コード文字は，連続コードの256からが割り振られています。したがって内部コード 8140 は，連続コードの 256 に相当します。内部コードの終端 FCFC に相当する連続コードは，11535 です。内部コードの EFFC（連続コードの 9091）までが，JIS 情報交換用漢字符号系に相当します。

連続コードと内部コードの関係



b1，b2は，内部コードの第1，第2バイトを表わす

JIS区点コードと内部コードの関係

JIS 区点コードの始まり 0101 は，内部コードの始まり 8140 と一致しています。内部コードの 819E まで来ると JIS 区点コードの第 1 区が 0194 で終了します。続いて内部コードの 819F から 2 区の 0201 が始まり，内部コード 81FC で 2 区の 0294 が終了します。これが 94 区まで続きます。

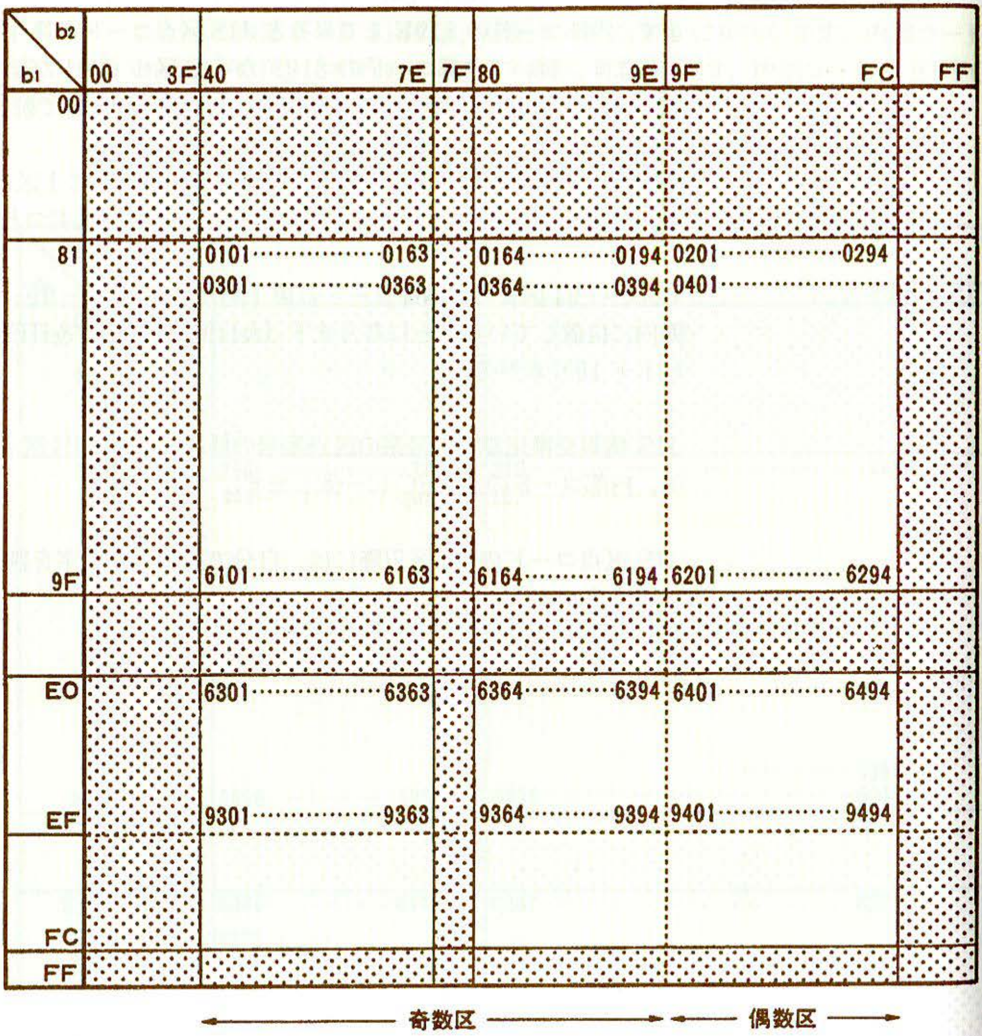
内部コードの第 1 バイトは，第 1 バイトが 81 のときに 1 区と 2 区，82 のときに 3 区と 4 区というように，JIS 区点コードの 2 区ずつと一致します。

1 区から 94 区までの内部コードの第 1 バイトは，81 ～ 9F，E0 ～ EF の範囲内に位置していることになります（&H9F-&H80，&HEF-&HDF=16，(31 + 16) ＊2=94)。

JIS 情報交換用漢字符号系の区点番号の終端である JIS 区点コードの 9494 は，内部コードの EFFC に一致します。

JIS 区点コードの 95 区以降には，自分で定義した文字を割り当てることができます。

JIS区点コードと内部コードの関係



b1，b2は内部コードの第1，第2バイトを表わす

JIS16進コードと内部コードの関係

JIS16進コードは，JIS区点コードの上2桁（区番号に相当）および，下2桁（点番号に相当）の01～94がそれぞれ，JIS16進コードの21～7Eに対応しています。

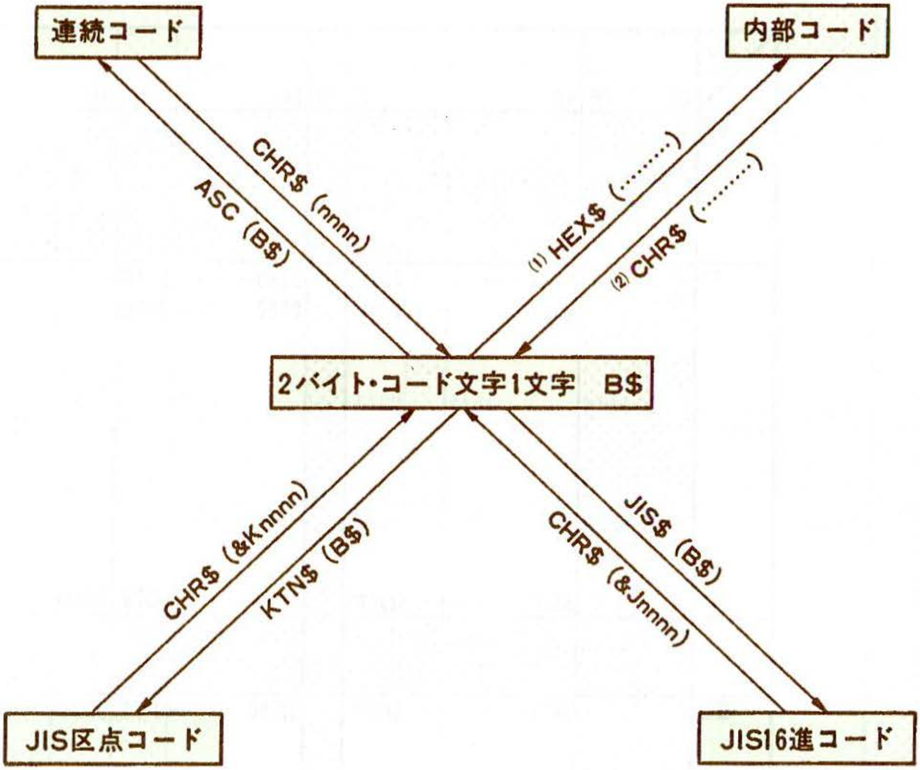
JIS16進コードと内部コードの関係

b1 \ b2	00 … 3F	40 … 7E	7F	80 … 9E	9F … FC	FF
00						
81		2121……215F 2321……235F		2160……217E 2360……237E	2221……227E	
9F		5D21……5D5F		5D60……5D7E	5E21……5E7E	
E0		5F21……5F5F		5F60……5F7E	6021……607E	
EF		7D21……7D5F		7D60……7D7E	7E21……7E7E	
FC						
FF						

b1，b2は内部コードの第1，第2バイトを表わす

コード間の変換

2バイト・コード文字における連続コード，内部コード，JIS区点コード，JIS16進コードは，次の関数を用いて変換することができます。B$ は2バイト・コード文字一文字を表しています。nnnn は各コードを表します。



(1) HEX$(ASC (LEFT$ (B$, 1)))+HEX$(ASC (RIGHT$ (B$, 1)))

(2) CHR$(VAL ("&H"+LEFT$ ("nnnn", 2)), VAL ("&H"+RIGHT$ ("nnnn", 2)))

次のプログラムは前述のコード間の変換式を用いた例です。

```
100 '*** コード変換プログラム ***
110  DIM A$(5)
120 '--- 入力画面の表示 ---
130 SCREEN 0 : KEY OFF : CLS
140 PRINT "*** コード変換 ***" : PRINT
150 A$(1)="連続コード"
160 A$(2)="内部コード"
170 A$(3)="JIS区点コード"
180 A$(4)="JIS16進コード"
190 A$(5)="文字"
200 FOR I = 1 TO 5
210   PRINT " [";I;"]  ";A$(I)
220 NEXT I
230 '--- データの入力 ---
240 PRINT : PRINT "[入力コード],[出力コード],";
245 PRINT " [コードまたは文字]"
250 PRINT "の形式で入力してください。"
255 INPUT "===> ",ICODE,OCODE,CODE$
260 '--- データのチェック ---
265 IF ICODE<1 OR ICODE>5 THEN 610
270 IF OCODE<1 OR OCODE>5 THEN 610
280 ICODE$ = A$(ICODE) : OCODE$ = A$(OCODE)
290 IF ICODE<>5 THEN CODE$=CSNG$(CODE$)
300 '*-- 入力コード別処理(文字イメージに変換) --*
310 ON ICODE GOTO 330,350,380,400,420
320 GOTO 610
330 '--- 連続コードを
340 C$=CHR$(VAL(CODE$)) : GOTO 450
350 '--- 内部コードを
360 IF LEN(CODE$)<>4 THEN 610
370 C$=CHR$(VAL("&H"+LEFT$(CODE$,2))
,VAL("&H"+RIGHT$(CODE$,2))) : GOTO 450
380 '--- JIS区点コードを
```

```
390 C$=CHR$(VAL("&K"+CODE$)) : GOTO 450
400 '--- JIS16進コードを
410 C$=CHR$(VAL("&J"+CODE$)) : GOTO 450
420 '--- 文字列を
430 C$=LEFT$(CODE$,2)
440 IF LEN(C$)<>2 OR KLEN(C$)*2<>LEN(C$) THEN 610
450 '*-- 出力コード別処理 --*
460 ON OCODE GOTO 480,500,520,540,560
470 GOTO 610
480 '--- 連続コードに
490 O$=STR$(ASC(C$)) : GOTO 580
500 '--- 内部コードに
510 O$=HEX$(ASC(C$)) : GOTO 580
520 '--- JIS区点コードに
530 O$=KTN$(C$) : GOTO 580
540 '--- JIS16進コードに
550 O$=JIS$(C$) : GOTO 580
560 '--- 文字列に
570 O$=C$
580 '--- 結果の表示 ---
590 PRINT ICODE$;"";CODE$;"の";OCODE$;"は";O$;"です。"
595 END
600 '--- 誤り入力の処理 ---
610 BEEP : PRINT
615 PRINT "***入力データに誤りがあります。";
620 PRINT "もう一度入力してください。***" : GOTO 230
```

拡張コード

拡張コードは，連続コードでは識別できないキー（カーソル移動キーなど）やキーを組み合わせて1つの機能とする場合（Ctrl キー＋ ← キーなど）のキー入力を識別するためのコードです。

拡張コードは，INKEY$ 関数で調べます。拡張コードは次の形式の2文字ストリングです。

空ストリング＋コード

空ストリングはコードの0です。2バイト目のコードには次表のコードが割り振られています。したがって，プログラムでこれらのキーの操作を識別するには，拡張コードであることを1バイト目で判断し，2バイト目でどのキー操作かを判断します。

次のプログラムは，拡張コードを使用した例です。

```
100 SCREEN 0 : KEY OFF : CLS : PF12 = 0
110 PRINT "***** 拡張コードの使用例 *****"
120 PRINT "* 文字列を入力してください。    *"
130 PRINT "* カーソル移動キー←→が         *"
135 PRINT "* 使用できます。                *"
140 PRINT "* 終了するためには PF12 を      *"
145 PRINT "* 押してください。              *"
150 PRINT "* 改行キーを押すとカーソルが    *"
160 PRINT "* 最初の位置に戻ります。        *"
170 PRINT "**********************************"
180 LOCATE 15,3 : PRINT "===> "
190 LOCATE 15,8
200 GOSUB 5000
```

```
220 IF PF12 = 1 THEN END
230 GOTO 190
5000 '
5010 '--- キー操作を処理するルーチン ---
5020 '
5030 '--- INKEY$ でキーボード・バッファーより
5040 ' 入力データを読み取り，BUF$ に割り当てる ---
5050 '
5060 BUF$ = ""
5070 ROW = CSRLIN
5080 COL = POS(0)
5090 TEMP$ = INKEY$
5100 '--- 拡張コードの 2 バイト目を得る
5110 TEMP2$ = MID$(TEMP$,2,1)
5120 '--- PF12が押されたら終わる
5130 IF TEMP2$ <> CHR$(&H55) THEN GOTO 5160
5140 PF12 = 1 : RETURN
5150 '--- 改行キーが押されたら終わる
5160 IF TEMP$ = CHR$(13) THEN RETURN
5170 IF TEMP2$ <> CHR$(75) GOTO 5200
5180 '--- ←キーが押されたら，カーソルを左へ
5190 LOCATE ROW,COL-1 : GOTO 5070
5200 IF TEMP2$ <> CHR$(77) GOTO 5240
5210 '--- →キーが押されたら，カーソルを右へ
5220 LOCATE ROW,COL+1 : GOTO 5070
5230 '--- →←キー以外の拡張コードは無視する
5240 IF MID$(TEMP$,1,1) = CHR$(0) THEN GOTO 5090
5250 '--- 入力文字を表示し，項目単位に集める
5260 PRINT TEMP$; : BUF$ = BUF$ + TEMP$
5270 GOTO 5070
```

拡張コード	キー操作
15	[⇧]+[⇄]
16-25	[前面キー]+[Q],[W],[E],[R],[T],[Y],[U],[I],[O],[P]
30-38	[前面キー]+[A],[S],[D],[F],[G],[H],[J],[K],[L]
44-50	[前面キー]+[Z],[X],[C],[V],[B],[N],[M]
59-68	[PF1]～[PF10]
71	[↖]
72	[↑]
73	[前面キー]+[↑]
75	[←]
77	[→]
79	[前面キー]+[↖]
80	[↓]
81	[前面キー]+[↓]
82	[挿入]
83	[削除]
84-93	[⇧]+[PF1]～[PF10]
94-103	[Ctrl]+[PF1]～[PF10]
104-113	[前面キー]+[PF1]～[PF10]
114	[Ctrl]+[印刷]
115	[Ctrl]+[←]
116	[Ctrl]+[→]
117	[Ctrl]+[↖]
118	[Ctrl]+[↓]
119	[Clear]
120-131	[前面キー]+[1],[2],[3],[4],[5],[6],[7],[8], [9],[0], −, ^
132	[Ctrl]+[↑]
147	[実行]
152	[取消]
153	[割込み]
154	[終了]
156	[前面キー]+[←]
157	[前面キー]+[→]

付録E

技術情報

記憶域マップ

次のページに記憶域マップの図を示します。

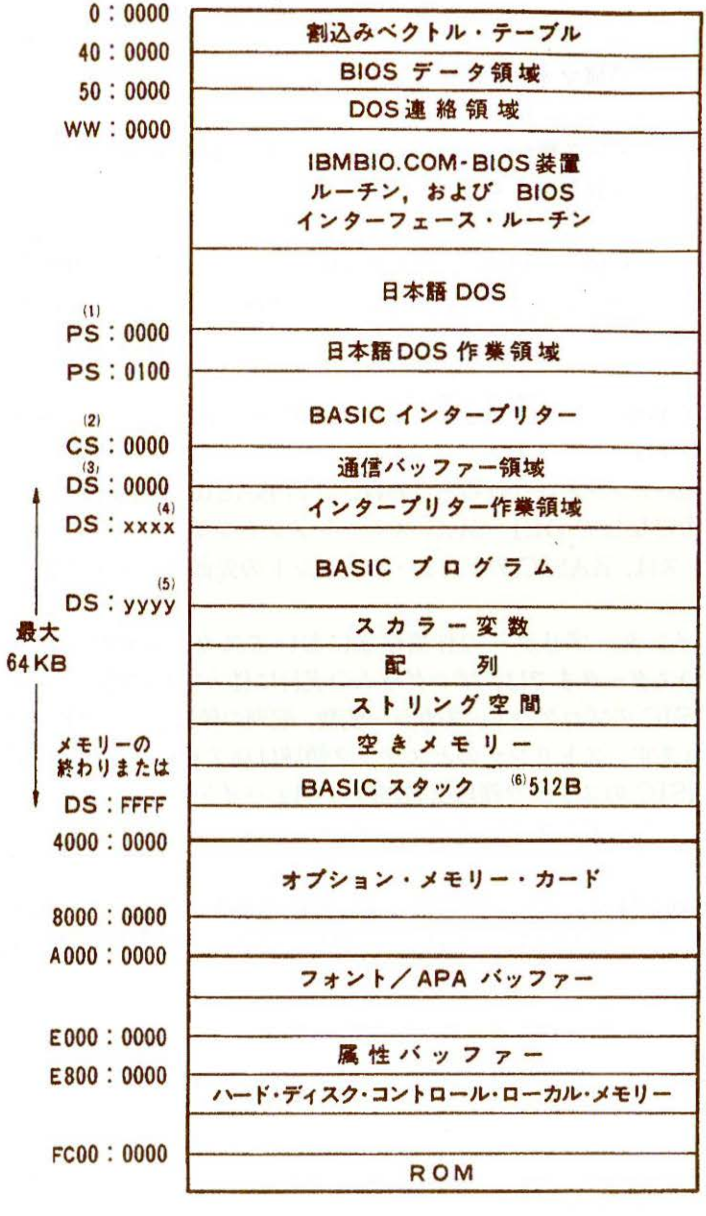
DOS は，低アドレス端から記憶位置を占めます。DOS の領域の大きさは，使用状況によって異なります。

通信バッファー以降，A0000 番地までは使用者のための任意使用領域です。そのため通信バッファーまでの領域を総じて簡略的に DOS の領域と呼ぶこともあります。

BASIC を始動すると通信バッファーのすぐ後に BASIC インタープリターの作業領域（最大 64K）が記憶域に割り振られます。通常 BASIC のデータ・セグメントと呼ばれるところは，この BASIC の作業領域です。アドレスの指定なしで DEF SEG ステートメントを実行すると，セグメント・アドレスは，BASIC のデータ・セグメントの先頭アドレスになります。

BASIC インタープリターの作業領域においてスカラー変数の領域から BASIC のスタックまでは，プログラムの実行に従って常に動的に配置されます。BASIC のプログラム，スカラー変数，配列の領域は，低アドレス側から配置されます。ストリングのとスタック領域は高アドレス側から配置されます。BASIC のスタック領域は省略時に 512 バイトですが，CLEAR ステートメントで変更できます。

A0000 番地以降のアドレスに対して POKE で書き込みを行わないでください。また，この領域に対して PEEK で値で読み取っても返る値は保証されません。



注）記憶域マップの中に付いている番号には，次の意味があります。

(1) PSはDOSプログラム・セグメントです。

(2) CSは通信バッファー・セグメントです（通信ポートの数，/C パラメーターによってサイズは変わります。）

(3) DSはBASICのデータ・セグメントです。

(4) 数xxxxはDS:30およびDS:31の位置にあります（底位バイト，高位バイト）。

(5) 数yyyyはDS:358，DS359の位置にあります。（底位バイト，高位バイト）。

(6) またはCLEARコマンドにより設定された値です。

変数の記憶方法

スカラー変数は，次のようにスカラー変数の領域に記憶されます。

バイト 0	1	2	3	4	4＋長さ
型	名前				データ
	文字	文字	長さ	長さで指定された個数の文字	2,3,4,8 バイト

型は変数の型を表す次の値が入っています。

2　整数
3　ストリング
4　単精度
8　倍精度

名前は変数名です。名前の最初の 2 文字が 1 バイト目と 2 バイト目に記憶されます。3 バイト目には変数名があと何文字あるかを示す値が入っています。変数名の先頭 2 文字を除く残りの文字は，4 バイト目以降に記憶されます。

変数名が 1 文字または 2 文字の場合この領域に 3 バイトとられます。3 文字以上の名前の場合，名前の文字数を n とすると n + 1 バイトの領域を占めます。

データは，型の値に応じて 2，3，4，8 バイトのいずれかの長さの値が入ります。VARPTR 関数は，この**データ**の領域を指すアドレスを返します。

ストリング変数の記憶　変数がストリング変数の場合，データにはストリングの情報を持ったストリング記述子が入っています。数値変数のように，**データ**に入っている値が実際の変数の値でないことに注意してください。

ストリング記述子の内容は，次のとおりです。

0バイト目	1バイト目	2バイト目
ストリングの長さ（0～255）	ストリングのオフセット 下位8ビット	ストリングのオフセット 上位8ビット

数値変数の記憶

数値変数の場合データには，次の形式で実際の数値が入ります。

整数変数

0バイト目	1バイト目
低位バイト	高位バイト

単精度変数

4バイトの中にBASICの内部浮動小数点2進形式で記憶されます。

倍精度変数

8バイトの中にBASICの内部浮動小数点2進形式で記憶されます。

BASICファイル制御ブロック

BASICファイル制御ブロックのアドレスは，VARPTRにファイル番号を指定すると得ることができます。アドレスは，BASICのデータ・セグメント内のオフセットです。ファイル制御ブロックに含まれている情報は次のとおりです。オフセットは，VARPTRで返される値を基準としています。

オフセット（バイト）	長さ（バイト）	内容
0	1	1－ 入力専用 2 － 出力専用 4 － ランダム 16 － アペンド専用 32 － 内部使用 128 － 内部使用
1	38	DOSファイル制御ブロック
39	2	順次ファイルは読み/書きしたセクター数 ランダム・ファイルは読み/書きしたセクター数＋1
41	1	読み/書き時のセクター内バイト数
42	1	入力バッファーに残っているバイト数
43	3	（予約済）
46	1	装置番号 0，1，2－ドライブ A:，B:，C: 248 － LPT3: 249 － LPT2: 250 － COM2: 251 － COM1: 252 － （予約済）

		253 − LPT1: 254 − SCRN: 255 − KYBD:
47	1	装置幅
48	1	PRINT#のバッファー内の位置
49	1	LOADとSAVEが内部使用 (データ・ファイル用には使用されない)
50	1	タブ拡張中に使用される出力位置
51	128	物理的データ・バッファー DOSとBASIC間データ転送に使用 順次入出力のときにデータの検査が可能
179	2	可変長レコード・サイズ(省略時128バイト) OPENステートメントで設定されたもの
181	2	現在の物理レコード番号
183	2	現在の論理レコード番号
185	1	(予約済)
186	2	ディスク・ファイル専用 (PRINT#, INPUT#, WRITE#の位置)
188	n	FIELDデータ・バッファー 大きさnはBASICコマンドの/S:で決まる(ランダム入出力のときにデータの検査が可能)

キーボード・バッファー

キーボードでタイプされた文字は処理されるまでキーボード・バッファーに保持されます。バッファーには最大 31 文字（31 バイト）まで保持されます。31 文字より多くタイプしようとすると警報音が鳴ります 。

INKEY$ 関数は，キーボード・バッファーに処理待ち状態の文字が複数ある場合も 1 文字のみ読み取ります。これに対し INPUT$ 関数は複数の文字を読み取ることができます。ただし，INPUT$ 関数の場合は，要求した数の文字がバッファーに存在しない場合指定した数の文字がタイプされるまで待っています。

システム・キーボード・バッファーは，次のプログラム行を使って消去することができます。

```
DEF SEG=0:POKE 1050,PEEK (1052)
```

INKEY$ や INPUT$ で実際の処理を行う前にキーボード・バッファーを消去する（BASIC の行バッファーを消去する）には，次のように記述します。

```
DEF SEG : POKE 106,0
```

印刷制御文字の指定

ここでは，BASIC からプリンターに制御文字を送る方法について説明します。

BASIC からは，LPRINT ステートメントを使用して制御文字をプリンターに送ります。この制御文字は，プリンターによって異なりますので，指定できるすべての制御文字の詳細は，各プリンターの操作ガイドを参照してください。ここでは，プログラム例で使用した制御文字についてのみ後述します。

制御文字は，LPRINT ステートメントで次のように指定します。

```
LPRINT "ABCDE";CHR$(&HA)
```

このステートメントは，印刷する文字 ABCDE の後ろに改行文字を付けてプリンターに送ります。&HA は，改行を意味する制御文字です。制御文字のみ送るときは，LPRINT CHR$ (&HA) と記述します。また，2 バイトの制御文字を送る場合は，CHR$ を＋（プラス）でつないで記述するかまたは，LPRINT CHR$ (&H1B,&H5D) のように記述します（CHR$ 内で記述する制御文字には 10 進数，16 進数が使用できます。16 進数の場合は，値の先頭に&H を付けます）。

LPRINT ステートメントを使用する場合は，次の点に注意してください。

- LPRINT の後に付けることができるセミコロン(;)は，復帰/改行を抑止する働きをしています。LPRINT "ABCD";と記述すると次に印刷する文字は，次の行の先頭からではなく ABCD のすぐ後から印刷されます。
- WIDTH"LPT1:"で変更しない限り LPRINT は，出力幅を 80 と想定しています。

BASICは通常CHR$（&H20）以上の文字を印刷可能文字，それ以下の文字を印刷不可能文字として扱います。これに関して以下の点に注意して下さい。

- 制御文字列中に&H20以上の文字が含まれている場合これらは印刷されないにもかかわらず，BASICは印刷可能文字と認識するため，TAB（n），などの指定が左にずれる場合があります。またWIDTHステートメントで指定した幅にいたらないうちに改行が行われる場合があります。

- 逆に&H20以下で印刷可能な文字（CHR$（1）…など）を指定すると上記と反対にTAB（n），の指定が右にずれたり，WIDTHで指定した幅になっても改行が行われない場合があります。

プログラム例 1

次の例は，縦方向 12 インチの連続用紙を使用した例です。

```
10 LPRINT CHR$(&H1B,&H46,0,&H48);
20 FOR I= 1 TO 3 : LPRINT : NEXT
30 FOR I= 1 TO 66
40 LPRINT I
50 NEXT
60 LPRINT CHR$(&HC);
70 GOTO 10
```

行番号 10 は，ページ長さを設定しています。ページ長さは，次の形式で指定します。

1B　46　n1　n2

1B　46 は 16 進数です。これは，ページ長さを示す文字です。n1 と n2 には，1 ページの長さを設定します。n1　n2 の単位は，4.23mm（1/6 インチ）で，範囲は 0001 から 01FF（16 進数）です。 0001 を指定する仮定した場合 n1 には 00 を n2 には 01 を指定します。 この例の場合，縦 12 インチを 1 ページとしているため n1　n2 は 00　48（16 進数）となります。

このページ指定の場合，1 ページに 72 行印刷できますが，ページの最初と最後は 3 行ずつ空送りしています。行番号 20 は，ページの先頭で 3 行送る指定です。

行番号 60 では，書式送りをしています。書式送りの制御文字を送ると印字バッファーにあるデータがすべて印字され，最初に用紙を設定したページの先頭まで送られます。

プログラム例 2

LLIST コマンドは，記憶域にあるプログラムのリストを印刷しますが，このコマンドを使用して,連続用紙にミシン目を避けて印刷することはできません。これを行うためには，プログラムを文字形式ファイルとして保管し(SAVE　"XXXX"，A を使用して。XXXX は，プログラム名)，1 行ずつ読みながら印刷します。次のプログラム例は，この処理を行ったものです。用紙は，プログラム例 1 で使用したものと同じものを想定しています。また，1 行の長さは，79 バイト以下であると仮定しています。

このプログラム例では，横方向の文字数に関する処理をしていませんが，1 行の文字数が，80 バイト以上になる場合は，WIDTH ステートメントを使用して文字幅を変更してください。またデータの 1 行の文字数が用紙の 1 行の文字数より多くなる場合は，そのための処理も必要になります。

```
5 '*** プログラム・ファイルの印刷 ***
10 CNT = 1
20 '--- ファイル名の入力 ---
30 INPUT "ファイル名は"; FIL$
40 '--- ファイルのオープン ---
50 OPEN FIL$ FOR INPUT AS #1
60 '--- 印刷処理 ---
70 LPRINT CHR$(&H1B,&H46,0,&H48);
80 '----------------ページの大きさの設定
90 LPRINT : LPRINT : LPRINT
100 '---------------用紙の先頭 3 行あける
110 LINE INPUT #1,FLINE$
120 '---------------文字形式ファイルの 1 行を読む
130 IF  EOF(1) THEN END
140 '---------------データが無ければ終了
150 LPRINT FLINE$
160 '---------------1 行を印刷
170 CNT = CNT + 1
180 IF CNT <> 67 THEN GOTO 110
190 '---------------1 頁に 66 行のデータを印刷
200 LPRINT CHR$(&HC);
210 '---------------FFを送る
220 GOTO 90
```

付録F

グラフィックス

BASICのグラフィックス

ここでは，BASIC のグラフィックスの機能を紹介することを目的として，次の 3 つのグラフのプログラム例を挙げて説明します。

- 円グラフ
- 棒グラフ
- 絵グラフ

これらのプログラム例は，グラフィックスの機能の紹介が目的であるため，複雑な機能は盛り込んでいません。

これらのプログラムは，16 ドット・システム/24 ドット・システム,単色モデル/カラー・モデルのいずれのモデルでも実行できるように配慮しています。

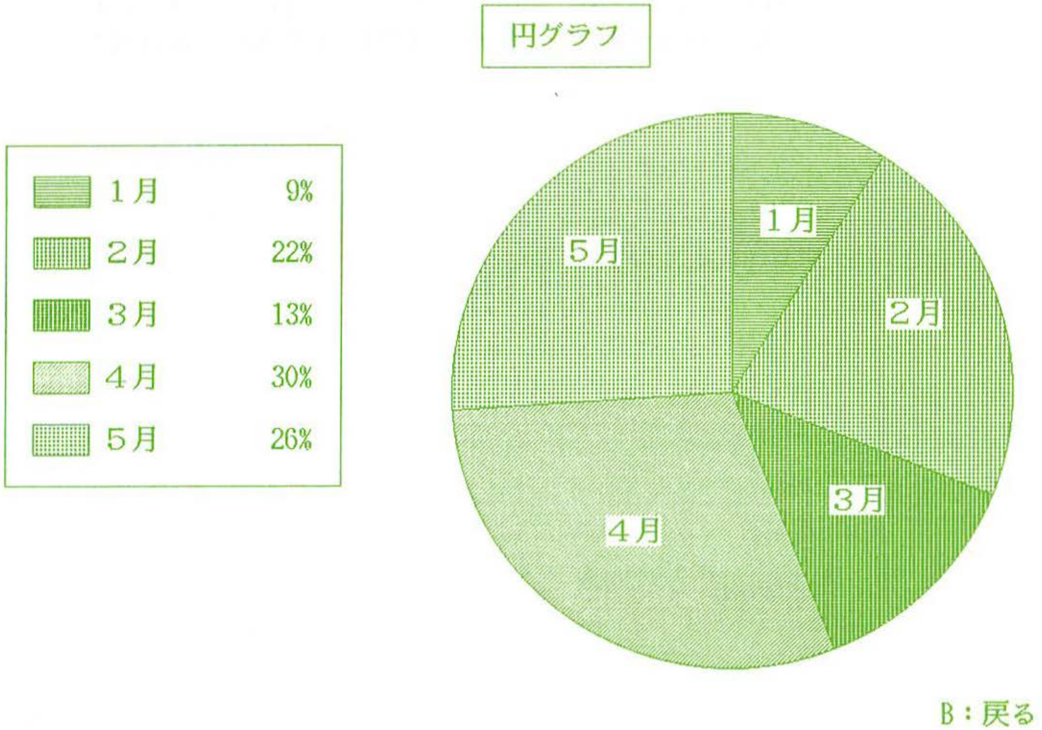
円グラフ

「**円グラフ**」作成プログラムを紹介します。このプログラムは，入力データから百分率を求め，模様で円グラフの各面を塗って表示します。したがって，プログラムはデータ入力，百分率の計算，円グラフの出力の各部分からなります。ここでは，データの個数を5つまでに限定しています。また，円グラフ内に表示しているデータ項目名は重なるか否かを調べていません。

プログラムの目的

- 入力されたデータの百分率を求め円グラフを表示する。

出力例



プログラム・リスト

```
10 '--------------
20 T4$="円グラフ
30 '--------------
40 '
50 '--- システムを調べる ---
60 '
70 KEY OFF : LOCATE,,0 : SCREEN 2,1
80 '--- カラー? 単色? ---
90 CFLG = 0   '--- CFLG(=0:単色,=1:カラー)
100 PSET(0,0),15 : RET = POINT(0,0)
110 IF RET = 15 THEN CFLG = 1
120 '--- 16ドット? 24ドット? ---
130 SFLG = 0  '--- SFLG(=0:24,=1:16)
140 SFLG = ABS(POINT(1023,767))
150 '
160 '--- 変数の初期設定 ---
170 '
180 DIM ELM$(5)      'データ名
190 DIM ODAT(5)      '基データ
200 DIM PER%(5)      '百分率
210 DIM RAD1(5)      'ラジアン
220 DIM RAD2(5)      '基準位置からのラジアン
230 DIM T$(5)        'グラフィックの模様
240 DIM CPX(5)       '項目名を表示するX位置
250 DIM CPY(5)       '項目名を表示するY位置
260 N = 0            'データ数
270 PI = 3.141593    'π
280 IF SFLG = 0 THEN DX = 13 : DY = 29   '24ドット・システム
290 IF SFLG = 1 THEN DX =  9 : DY = 20   '16ドット・システム
300 '
310 '--- 画面の初期設定 ---
320 '
330 IF CFLG = 1 THEN C = 3 : COLOR 0,0,0 : ELSE C = 1    '表題の表示
```

```
340 LINE(DX*34,DY/2)-(DX*46,DY*2+DY/2),C,B
350 LOCATE 2,37 : PRINT "円グラフ";
360 '
370 '--- データの入力 ---
380 '
390 VIEW PRINT 5 TO 23                    'データ入力領域
400 MSG = 0  : DSUM = 0 : LOCATE,,1
410 INPUT "[項目数](5以内)==> ",N
420 IF N > 5 THEN BEEP : GOSUB 600 : CLS 2 : GOTO 400
430 PRINT : PRINT " [項目名,データ] (項目名:10バイト以内,データ:半角数字)"
440 PRINT
450 PRINT "(例:1 月,30)" : PRINT
460 FOR I = 1 TO N
470   INPUT "==> ",ELM$(I),ODAT(I)
480   DSUM = DSUM + ODAT(I)
490 NEXT I
500 VIEW PRINT 5 TO 24 : CLS 2 : LOCATE,,0 : GOTO 680
510 '
520 '--- 百分率を求める ---
530 '
540 PSUM% = 0
550 FOR I = 1 TO N
560   PER%(I) = ODAT(I)/DSUM * 100
570   PSUM%   = PSUM% + PER%(I)
580 NEXT I
590 TEST = 100 - PSUM% : PER%(N) = PER%(N) + TEST : GOTO 680
600 '
610 '--- メッセージ処理ルーチン ---
620 '
630 VIEW PRINT 24 TO 24 : CLS 2
640 LOCATE 24,1
650 PRINT "項目は5つ以内です。もう一度入力してください。";
660 VIEW PRINT  5 TO 23
670 RETURN
680 '
```

```
690 '--- 百分率から角度(ラジアン)を求める ---
700 '
710 FOR I = 1 TO N
720   RAD1(I) = PER%(I)*2*PI / 100
730 NEXT I
740 '
750 '--- 基準位置からの角度ラジアンに直す ---
760 '
770 FOR I = 1 TO N
780   FOR J = 1 TO I
790     RAD2(I) = RAD2(I) + RAD1(J)
800   NEXT J
810 NEXT I
820 '
830 '--- 円グラフを描く ---
840 '
850 CX = DX*52 : CY = DY*13 : R = DY*9
860 CIRCLE (CX,CY),R                    '円を描く
870 IF N = 1 THEN 970
880 FOR I = 1 TO N
890   X =  SIN(RAD2(I))*R + CX           '円と仕切り線の
900   Y = -COS(RAD2(I))*R + CY           '    接点の座標
910   LINE (CX,CY)-(X,Y)                '中心から接点へ
920 NEXT I
930 '
940 '--- 区分ごとに塗りつぶす ---
950 '
960 '--- 16色カラー・グラフィック用の模様
970 IF CFLG = 0 THEN 1040
980 T$(1)=CHR$(&H33,&H33,&H33,&H33,0,0,0,0,0,0,0,0)    '横縞
990 T$(2)=CHR$(&H4,0,&H4,0,&H44,&H40,&H44,&H40,&H4,0,&H4,0,0,0,0,0) '格子
1000 T$(3)=CHR$(&H5,&H55,&H5,&H55,&H5,&H55,&H5,&H55,0,0,0,0)  '箱
1010 T$(4)=CHR$(&H8,0,&H8,0,&H80,0,&H80,0,0,&H8,0,&H8,0,&H80,0,&H80) '斜網
1020 T$(5)=CHR$(0,0,0,0,&HF,&HF0,&HF,&HF0,&HF,&HF0,&HF,&HF0,0,0,0,0) '箱
1030 GOTO 1100
```

```
1040 '--- 単色グラフィック用の模様
1050 T$(1)=CHR$(&HFF,0,0)   '横縞
1060 T$(2)=CHR$(&H44,&HEE,&H44,0) '格子
1070 T$(3)=CHR$(&H77,&H77,0,0)  '箱
1080 T$(4)=CHR$(&H44,&H88,&H11,&H22) '斜網
1090 T$(5)=CHR$(0,&H66,&H66,0) '箱
1100 '
1110 FOR I = 1 TO N
1120   RAD = RAD2(I)-RAD1(I)/2
1130   CPX(I)   =  SIN(RAD)*R/3*2 + CX
1140   CPY(I)   = -COS(RAD)*R/5*3 + CY
1150   PAINT (CPX(I),CPY(I)),T$(I)
1160 NEXT I
1170 '
1180 '--- 項目名を表示する ---
1190 '
1200 FOR I = 1 TO N
1210   ROW% = CPY(I)/DY : COL% = CPX(I)/DX   '文字位置を決める
1220   L  = LEN(ELM$(I))/2
1230   COL% = COL%-L+1
1240   LOCATE ROW%,COL%
1250   PRINT ELM$(I)
1260 NEXT I
1270 '
1280 '--- 項目名と百分率を表示する ---
1290 '
1300 IF CFLG = 1 THEN C = 3 ELSE C = 1
1310 LINE(0,5*DY)-(24*DX,16*DY),C,B
1320 IF CFLG = 1 THEN C = 15 ELSE C = 1
1330 FOR I = 1 TO N
1340   X1 = DX*2 : Y1 = DY*4+DY*2*I
1350   X2 = DX*6 : Y2 = DY*5+DY*2*I
1360   LINE (X1,Y1)-(X2,Y2),C,B
1370   PAINT (X1+5,Y1+5),T$(I)
1380   LOCATE 5+2*I,8 : PRINT USING "&        & ###%";ELM$(I);PER%(I)
```

```
1390 NEXT I
1400 '
1410 '--- キー入力の処理
1420 '
1430 LOCATE 24,68 : PRINT "E:終了";
1440 POKE 106,0
1450 K$ = INKEY$ : IF K$ = "" THEN 1450
1460 IF K$ = "E" OR K$ = "e" THEN SCREEN 0,0 : COLOR 0,0,0 : LOCATE,,1 : END
1470 GOTO 1450
```

プログラムの処理内容

システムを調べる

このプログラムでは，まず最初に「**このプログラムが実行されているハードウェアについて**」調べています。

1 **カラーのモデルであるか，単色のモデルであるか**

BASIC の画面モードはSCREEN 2,1を指定するとカラーのモデルでは，カラー・グラフィック・モードに，単色のモデルでは，単色グラフィック・モードになります（旧 C モデルのみ例外で，単色グラフィック・モードになります)。POINT 関数は画面上に描かれた点のパレット番号（単色の場合は 1 または 0）を返します。これら機能を用いるとどちらのモデルか，調べることができます。

```
70 KEY OFF : LOCATE,,0 : SCREEN 2,1
80 '--- カラー? 単色? ---
90 CFLG = 0    '--- CFLG(=0:単色,=1:カラー)
100 PSET(0,0),15 : RET = POINT(0,0)
110 IF RET = 15 THEN CFLG = 1
```

2 **16 ドット・システムであるか，24 ドット・システムであるか**

16 ドット・システムと 24 ドット・システムでは画面上の点の下図が異なり

ます。POINT 関数は範囲外の座標値を指定すると-1 を返します。したがって，16 ドット・システムでは範囲外となるような座標値を指定すれば，どちらのシステムか調べることができます。

```
120 '--- 16ドット?  24ドット? ---
130 SFLG = 0   '--- SFLG(=0:24,=1:16)
140 SFLG = ABS(POINT(1023,767))
```

変数の初期設定

プログラム内で使用する変数や配列の初期値を設定すると共に，文字ボックスの X 方向の値，Y 方向の値をシステムに合わせて設定しています。

```
280 IF SFLG = 0 THEN DX = 13 : DY = 29   '24ドット・システム
290 IF SFLG = 1 THEN DX =  9 : DY = 20   '16ドット・システム
```

データの入力

データ入力には INPUT を使用しています。この部分では，データ入力部分以外に表示されているものを，後で消去せずにすむよう VIEW PRINT を使用してデータ入力領域を定めています。VIEW PRINT で範囲を指定すると画面はその範囲のみでスクロールを行います。そのため，範囲外に表示しているものを不必要にスクロールせずにすみます。

また一般的に，画面上の任意の場所でデータ入力を行うには，INKEY$ を用いますが，カーソルの移動キーや［挿入］キーや［削除］キーなどデータ編集用のキーは，すべて独自の処理をしなければなりません。したがって，処理が繁雑になります。これに比べて INPUT は限られた範囲でありながらデータ編集用のキーの処理を行っていますので，容易に使用することができます。

INPUT は入力したデータとそのデータを割り当てる変数との間に矛盾があると**次の行の先頭に**?Redo from start（始めからやり直し）というメッセージを表示し，データの入力を求めます。したがって，画面の任意の位置で，INPUT を用いて入力作業をしようとしても入力を誤ると再び任意の位置で作業することはできなくなります。このため，このプログラムはデータ入力域を VIEW PRINT で限定し，画面の左端を使用して入力作業を行っています。

同様に，入力データに対するメッセージもメッセージ領域を VIEW PRIN

Tで24行目に定めて表示しています。

百分率を求める

入力データの総計は，データ入力と同時に求めています。百分率を整数値で得るにあたり，実数値が四捨五入されて百分率の合計が100にならない場合の処理として，便宜的に最終データを加工しています。

角度を求める

百分率に2 π/100をかけると角度（ラジアン）が求まります。ただし，円グラフなので，基準となる位置からの角度（ラジアン）を求める必要があります。

円グラフを描く

円の中心からある角度の仕切り線を引くには，プログラム・リストのように三角関数で接点の値を求めます。

```
820 '
830 '--- 円グラフを描く ---
840 '
850 CX = DX*52 : CY = DY*13 : R = DY*9
860 CIRCLE (CX,CY),R                  '円を描く
870 IF N = 1 THEN 970
880 FOR I = 1 TO N
890   X =  SIN(RAD2(I))*R + CX        '円と仕切り線の
900   Y = -COS(RAD2(I))*R + CY        '     接点の座標
910   LINE (CX,CY)-(X,Y)              '中心から接点へ
920 NEXT I
```

座標値を指定して図を描くときは，事前に定義した文字ボックスの値から座標値を求めて指定しています。このように処理すると，システムの違いを吸収するだけでなく，ビジネス・グラフのように図形と文字が画面に混在する場合に，文字と図形の位置を合わせやすいという利点があります。

円グラフを塗る

円グラフの各部分の内側の座標を求めPAINTで模様を塗ります。模様は16色カラー用と単色用の2とおりを用意し，モデルにより使いわけています。

項目名を表示する

円グラフ内に項目名を表示するために，表示する文字列を調べています。表

示位置は円グラフを塗ったときの開始点を使用しています。

項目名と百分率を表示 円グラフの横に項目名と百分率を表示しています。

プログラムの終り キー入力を待ち，「E」が押されたら画面を文字モードにし，COLOR で指定された属性を元に戻し，カーソルを再び表示してプログラムを終了しています。

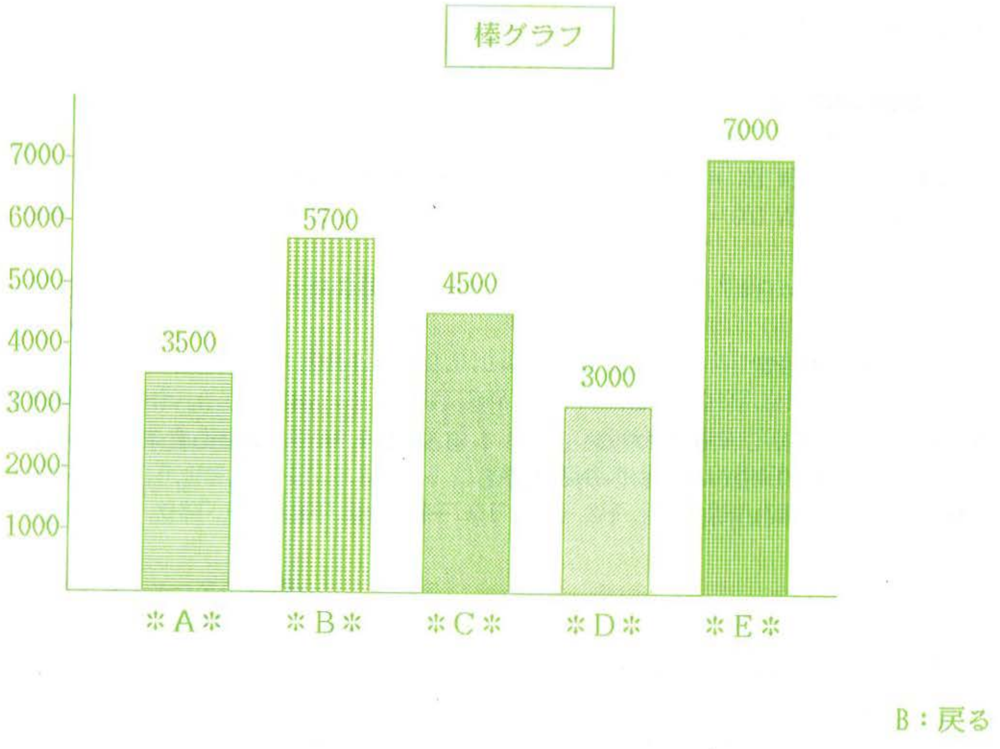
棒グラフ

「**棒グラフ**」作成プログラムは，一つデータを入力するごとに 1 本の棒を表示していきます。データの許容範囲は 0 ～ 7000 までで，それ以上のデータは 7000 までの長さと同じものを表示します。

プログラムの目的

● 入力データの値に従って棒グラフを表示する。

出力例



プログラム・リスト

```
10 '--------------
20 T5$="棒グラフ"
30 '--------------
40 '
50 '--- システムを調べる ---
60 '
70 KEY OFF : LOCATE,,0 : SCREEN 2,1
80 '--- カラー? 単色? ---
90 CFLG = 0   '--- CFLG(=0:単色,=1:カラー)
100 PSET(0,0),15 : RET = POINT(0,0)
110 IF RET = 15 THEN CFLG = 1
120 '--- 16ドット? 24ドット? ---
130 SFLG = 0  '--- SFLG(=0:24,=1:16)
140 SFLG = ABS(POINT(1023,767))
150 '
160 '--- 変数の初期設定 ---
170 '
180 IF SFLG = 0 THEN DX = 13 : DY = 29   '24ドット・システム
190 IF SFLG = 1 THEN DX =  9 : DY = 20   '16ドット・システム
200 '
210 DATA "*A*","*B*","*C*","*D*","*E*"
220 '
230 '--- 画面の初期設定 ---
240 '
250 IF CFLG = 1 THEN C = 9 : COLOR 0,0,0 : ELSE C = 1    '表題の表示
260 LINE(DX*34,DY/2)-(DX*46,DY*2+DY/2),C,B
270 LOCATE 2,37 : PRINT T5$;
280 '
290 '--- 軸を描く ---
300 '
310 IF CFLG = 1 THEN C = 15 ELSE C = 1
320 SC = 9 : SR = 20
330 PX = SC*DX - DX - DX/2 : PY = SR*DY - DY/2
```

```
340 LINE(PX,PY)-STEP(DX*55,0),C     'X軸
350 LINE(PX,PY)-STEP(0,-DY*16),C    'Y軸
360 '
370 '--- 項目を表示する ---
380 '
390 RESTORE 210
400 FOR I = 1 TO 5
410   LOCATE SR+1,SC+I*10-5,0
420   READ A$
430   PRINT A$;
440 NEXT I
450 '
460 '--- Y軸の目盛を表示 ---
470 '
480 FOR I = 7 TO 1 STEP -1
490   LOCATE SR-I*2,SC-5,0
500   PRINT USING "#000";I
510   LINE(PX,PY-I*2*DY)-STEP(-DX/2,0),C
520 NEXT I
530 '
540 '--- 模様 ---
550 '
560 '--- 16 色カラー・グラフィックの模様
570 IF CFLG = 0 THEN 640
580 T$(1)=CHR$(&H44,&H44,&H44,&H44,0,0,0,0,0,0,0,0)'横縞
590 T$(2)=CHR$(0,&H77,0,0,0,&H77,0,0,&H77,&H77,&H77,0,&H77,&H77,&H77,0)'格子
600 T$(3)=CHR$(&HAA,0,&HAA,0,&HAA,0,&HAA,0,0,&HAA,0,&HAA,0,&HAA,0,&HAA)'チェッカー
610 T$(4)=CHR$(&HE,0,&HE,0,&HE0,0,&HE0,0,0,&HE,0,&HE,0,&HE0,0,&HE0)'斜網
620 T$(5)=CHR$(&HF,&HFF,&HF,&HFF,&HF,&HFF,&HF,&HFF,&HF,&HFF,&HF,&HFF,0,0,0,0)'箱
630 GOTO 730
640 '--- 単色グラフィックの模様
650 T$(1)=CHR$(&HFF,&HFF,0,0) '横縞
660 T$(2)=CHR$(&H60,&H60,&HFC,&HFC)'格子
670 T$(3)=CHR$(&HCC,&HCC,&H33,&H33)'チェッカー
680 T$(4)=CHR$(&H44,&H88,&H11,&H22)'斜網
```

```
690 T$(5)=CHR$(&H77,&H77,&H77,0) '箱
700 '
710 '--- データの入力 ---
720 '
730 RESTORE 210
740 FOR I = 1 TO 5
750   READ A$
760   LOCATE ,,1
770   VIEW PRINT 23 TO 23
780   PRINT A$;"のデータは";
790   INPUT ; INDATA
800   IF INDATA>=0 THEN VIEW PRINT 1 TO 24 : GOTO 840
810   LOCATE ,,0
820   PRINT SPC(50);
830   GOTO 770
840 '
850 '--- 棒を描く ---
860 '
870   IF INDATA>7000 THEN YP = 70 : INY = 7000 ELSE YP = INT(INDATA/100) : INY =
INDATA
880   LOCATE SR-INT(YP/5+.5)-1,SC+I*10-6,0
890   PRINT USING "######";INDATA
900   X1 = (SC+I*10-3)*DX-3*DX-1 : X2 = X1 + 6*DX + 2
910   Y1 = INT(PY-(INY/500)*DY)  : Y2 = PY
920   LINE(X1,Y1)-(X2,Y2),C,B
930   CX = (X1+X2)/2 : CY = (Y1+Y2)/2
940   PAINT (CX,CY),T$(I)
950   VIEW PRINT 23 TO 23 : LOCATE 23,1,0
960   PRINT SPC(50);
970 NEXT I
980 '
990 '--- キー入力の処理
1000 '
1010 VIEW PRINT 1 TO 24 : LOCATE 24,68 : PRINT "E:戻る";
1020 POKE 106,0
```

```
1030 K$ = INKEY$ : IF K$ = "" THEN 1030
1040 IF K$ = "E" OR K$ = "e" THEN SCREEN 2,1 : COLOR 0,0,0 : LOCATE,,1 : END
1050 GOTO 1030
```

プログラムの処理内容

このプログラムの次の処理は，前述の「円グラフ」と同様なため，処理内容の説明は「円グラフ」を参照してください。

- システムを調べる
- 文字ボックスの値を持つ
- プログラムの終り

軸を描く

最初に棒グラフの原点となる位置を画面上の文字表示位置で定めます。図形はこの位置を基準にして文字ボックスの数で算出した座標点で描きます。LINE は開始点からの相対位置を指定しています。

```
280 '
290 '--- 軸を描く ---
300 '
310 IF CFLG = 1 THEN C = 15 ELSE C = 1
320 SC = 9 : SR = 20
330 PX = SC*DX - DX - DX/2 : PY = SR*DY - DY/2
340 LINE(PX,PY)-STEP(DX*55,0),C     'X軸
350 LINE(PX,PY)-STEP(0,-DY*16),C    'Y軸
```

このプログラムでは，9 桁目/20 行目を基準の位置にしています。

項目はあらかじめ DATA で用意しています。これを読んで表示しています。

また，Y 軸の目盛りは文字位置と線が合うように，文字ボックスの値を使用して座標指定しています。

データの入力

データの入力領域は VIEW PRINT で 23 行目に定めています。VIEW PRINT で入力領域を定める理由については「円グラフ」を参照してください。データ入力領域は PRINT で SPC を用いスペースを並べることにより適宜消去しています。

棒を描く

7000 以上のデータが入力されたときには棒の長さを 7000 の場合と同様にし，値は実際の値を表示するように入力データを調べて設定します。

棒には，すでに配列に用意しておいたデータを使用して PAINT で模様を塗ります。PAINT の開始点は，棒の中央の座標を求めて使用しています。

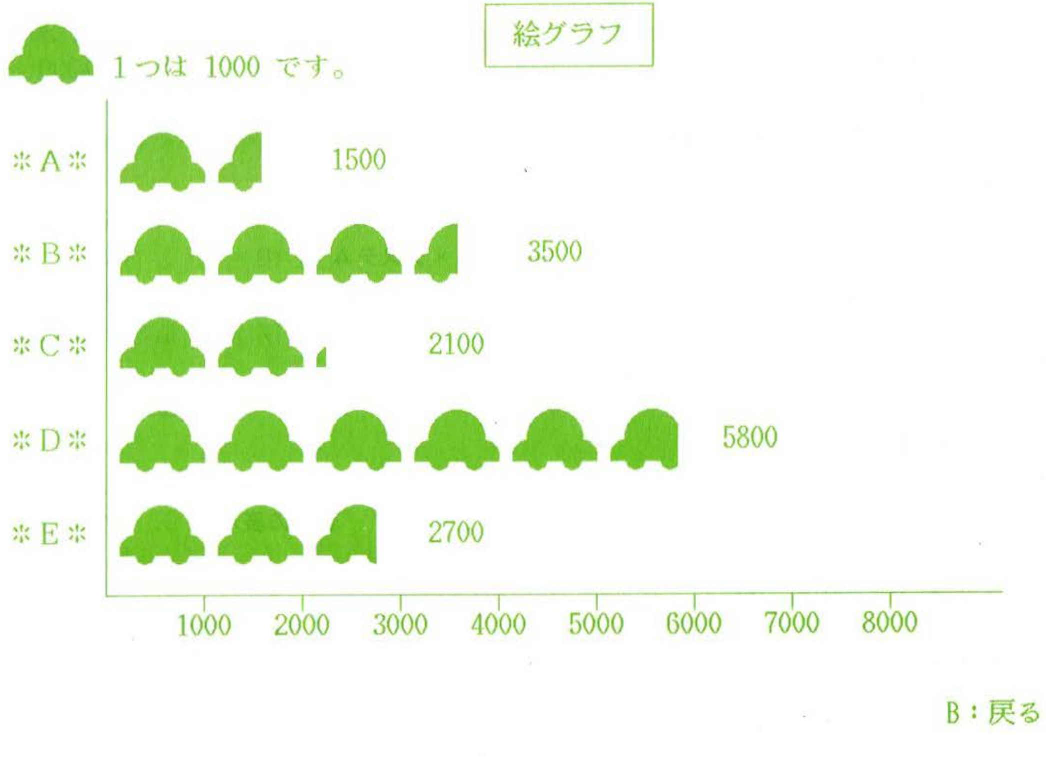
絵グラフ

ここでは，ある一つの図形を一単位として数量を表現するグラフを「**絵グラフ**」と呼びます。プログラム例として紹介する絵グラフは，車の 1 台を 1000 として，0 ～ 8000 までの入力データを表現します。このプログラムでは，画面のグラフ外の部分にまず車の絵を描きます。これを GET で入力データの値に従って配列に取り込んだのち，PUT で表示しています。

プログラムの目的

● 入力データの値に従って車の絵グラフを表示する。

出力例



プログラム・リスト

```
10 '--------------
20 T6$="絵グラフ"
30 '--------------
40 '
50 '--- システムを調べる ---
60 '
70 KEY OFF : LOCATE,,0 : SCREEN 2,1
80 '--- カラー?  単色?  ---
90 CFLG = 0    '--- CFLG(=0:単色,=1:カラー)
100 PSET(0,0),15 : RET = POINT(0,0)
110 IF RET = 15 THEN CFLG = 1
120 '--- 16ドット?  24ドット? ---
130 SFLG = 0  '--- SFLG(=0:24,=1:16)
140 SFLG = ABS(POINT(1023,767))
150 '
160 '--- 変数の初期設定 ---
170 '
180 DIM A%(2000)              '車の絵を保管する配列
190 FACT = 1
200 SC = 7  : SR = 20         '原点の基本となる行と桁
210 IF SFLG = 0 THEN DX = 13 : DY = 29   '24ドット・システム
220 IF SFLG = 1 THEN DX =  9 : DY = 20   '16ドット・システム
230 '
240 DATA "*A*","*B*","*C*","*D*","*E*"
250 '
260 '--- 画面の初期設定 ---
270 '
280 IF CFLG = 1 THEN C = 9 : COLOR 0,0,0 : ELSE C = 1     '表題の表示
290 LINE(DX*34,DY/2)-(DX*46,DY*2+DY/2),C,B
300 LOCATE 2,37,0 : PRINT T6$;
310 '
320 '--- 軸を描く ---
330 '
```

```
340 IF CFLG = 1 THEN C = 15 ELSE C = 1
350 PX = SC*DX : PY = SR*DY - DY/2 '原点
360 LINE(PX,PY)-STEP(DX*64,0) ,C    'X軸
370 LINE(PX,PY)-STEP(0,-DY*16),C    'Y軸
380 '
390 '--- 項目を表示する ---
400 '
410 RESTORE 240
420 FOR I = 1 TO 5
430   LOCATE SR-14+(I-1)*3,1,0
440   READ A$
450   PRINT A$;
460 NEXT I
470 '
480 '--- X軸の目盛を表示 ---
490 '
500 FOR I = 1 TO 8
510   LOCATE SR+1,SC+I*7-1
520   PRINT USING "#000";I
530   LINE(PX+I*7*DX,PY)-STEP(0,+DY/2),C
540 NEXT I
550 IF CFLG = 1 THEN C = 7 : C1 = 4 ELSE C = 1 : C1 = 1
560 CX = DX*3  : CY = DY*2
570 GOSUB 950
580 LOCATE 3,8,0 : PRINT "1つは 1000 です。";
590 '
600 '--- データの入力 ---
610 '
620 RESTORE 240
630 FOR I = 1 TO 5
640   READ A$
650   LOCATE ,,1
660   VIEW PRINT 23 TO 23
670   PRINT A$;"のデータは";
680   INPUT ; INDATA
```

```
690   LOCATE,,0
700   IF INDATA>=0 THEN VIEW PRINT 1 TO 24 : GOTO 740
710   LOCATE ,,0
720   PRINT SPC(50);
730   GOTO 660
740 '
750 '--- 絵を描く ---
760 '
770   IF INDATA>8000 THEN INX = 8000 ELSE INX = INDATA
780   CNT% = FIX(INX/1000) : LAST = INX MOD 1000 : IF LAST > 0 THEN ADD = 1 ELSE ADD
= 0
790   FACT = 1
800   LPY  = SR*DY-DY*3*(6-I)-DY
810   FOR J = 1 TO CNT%+ADD
820     LPX = SC*DX+DX+(J-1)*DX*7
830     IF J = CNT%+ADD AND ADD = 1 THEN FACT = LAST/1000
840     X  = CX-DX*3    : Y  = CY-DY*1.5
850     WX = DX*6*FACT  : WY = DY*3
860     GET(X,Y)-(X+WX,Y+WY),A%
870     PUT(LPX,LPY),A%
880   NEXT J
890   LOCATE SR-14+(I-1)*3,INT(LPX/DX)+8,0
900   PRINT INDATA;
910   VIEW PRINT 23 TO 23 : LOCATE 23,1,0
920   PRINT SPC(50);
930 NEXT I
940 GOTO 1180
950 '
960 '--- 車を表示するサブルーチン ---
970 '
980 '入力データ: 中心点      CX,CY
990 '              色        C ,C1
1000 '              表示範囲    FACT(0～1)
1010 '
1020 R = DX*2 : PI = 3.141593
```

```
1030 CIRCLE(CX,CY),R,C,2*PI-.1,PI+.1
1040 CIRCLE(CX-DX*2,CY+DY*.5),DX,C,PI/2,PI+.1
1050 CIRCLE(CX+DX*2,CY+DY*.5),DX,C,0+.1,PI/2
1060 X = CX-DX*3 : Y = CY+DY*.5
1070 LINE(X,Y)-STEP(0,DY*.2),C
1080 LINE -STEP(DX*6,0),C
1090 LINE -STEP(0,-DY*.2),C
1100 PAINT (CX,CY),C,C
1110 DDX = DX+DX*.2
1120 CIRCLE(CX-DDX,Y+DY*.2),DX*.7,C1 : PAINT (CX-DDX,Y),C1,C1
1130 CIRCLE(CX+DDX,Y+DY*.2),DX*.7,C1 : PAINT (CX+DDX,Y),C1,C1
1140 RETURN
1150 '
1160 '--- キー入力の処理
1170 '
1180 VIEW PRINT 1 TO 24 : LOCATE 24,68 : PRINT "E:戻る";
1190 POKE 106,0
1200 K$ = INKEY$ : IF K$ = "" THEN 1200
1210 IF K$ = "E" OR K$ = "e" THEN SCREEN 2,1 : COLOR 0,0,0 : LOCATE,,1 : END
1220 GOTO 1200
```

プログラムの処理内容

このプログラムの次の処理は，前述の「円グラフ」または「棒グラフ」と同様なため，これらの処理内容の説明は「円グラフ」「棒グラフ」を参照してください。

- システムを調べる
- 文字ボックスの値を持つ
- データ入力の流れ
- 軸の表示
- プログラムの終り

ただし，グラフの軸は「棒グラフ」の場合Y軸方向に目盛があるのに対して，この絵グラフでは，X軸方向に目盛を設定しています。すなわち，車の絵は，X軸方向に並べて描いています。

車を表示するサブルーチン

このサブルーチンは，次のデータを基にCIRCLE，LINE，PAINTを使用して車の絵を描きます。

- 車を表示する位置の座標
  車の絵はこの位置を中心点として描きます。

- 車の色

値に従って車を表示

入力データが1000単位でなく端数が生じる場合は，その端数に合わせた車の表示を行っています。すなわち，車の絵をGETで得るときに，車のX方向の長さを1とした場合の表示割合を0～1の値で表したものをX方向の全体の長さに乗じて，表示範囲を決めています。

車を任意の長さで表示する方法は，一旦すべての図形を配列に取り込んだ後加工するなど，他にも方法は考えられますが，ここでは容易にできる処理として，あらかじめ求める範囲で図形のデータを得る方法で行っています。

値に従って一列の車を表示する部分の処理の流れは次のようです。

1 8000以上の値が入力された場合は車の絵を8個表示し，値の表示は実際の値を表示するように設定する。

2 入力データを1000で割り，全体を表示する車の個数を求める。また，入力データを1000で割った場合の端数の有無を調べる。

3 まず，車の表示割合を1にしておく。この表示割合は常に絵のデータを得るときに使用されるが，最後の車を表示する場合に端数があれば表示割合を設定しなおす。

4 車の表示位置を求める。

5 表示割合にしたがって GET で車の絵を得る。

6 PUT で車の絵を表示する。

以上の処理はプログラム中の次の部分の処理の流れです。

```
770   IF INDATA>8000 THEN INX = 8000 ELSE INX = INDATA
780   CNT% = FIX(INX/1000) : LAST = INX MOD 1000 : IF LAST > 0
THEN ADD = 1 ELSE ADD = 0
790   FACT = 1
800   LPY  = SR*DY-DY*3*(6-1)-DY
810   FOR J = 1 TO CNT%+ADD
820     LPX = SC*DX+DX+(J-1)*DX*7
830     IF J = CNT%+ADD AND ADD = 1 THEN FACT = LAST/1000
840     X  = CX-DX*3    : Y  = CY-DY*1.5
850     WX = DX*6*FACT  : WY = DY*3
860     GET(X,Y)-(X+WX,Y+WY),A%
870     PUT(LPX,LPY),A%
880   NEXT J
```

付録G

用語集

アドレス：記憶位置。レジスター，記憶域の特定部分またはデータの宛先を指すもの。名前，ラベルまたは数で表現される。

アルゴリズム：ある問題を有限個の手続きで解くための，正確に定められた一連の規則または処理。

1バイト・コード文字：連続コード（文字に割り振られた10進数の番号）0～255で表現される文字セット。画面に表示したときに，半角で表示できる英数字，カタカナ，特殊文字の一部。

インターフェース：プログラムとプログラム，プログラムと機械などが情報を交換する際の接点となる境界区域。

ウインドー：グラフィック・モードで図形処理を行う場合に，使う人が想定する座標系内の取り出し領域。ワールド座標系の一部。

エコー：データを反響させること。通信では，受け取ったデータを送信側に返すこと。または，キー入力を文字として画面に表示させたり，プリンターに出力すること。

オーバーレー：プログラムのいろいろな段階で，内部で記憶している同一ブロックを繰り返し使用する方法。事前に読み込まれていたプログラムの一部を別のプログラムを読み込んで置き換えること。

オフセット：レコード，制御ブロック，記憶域内における開始位置からある位置までの変位。本システムのBASICではセグメント（記憶域内の64Kバイト以内の連続したブロック）内の変位を示すものとして用いることが多い。

オペランド：演算を行う対象。または，命令の一部分で処理や操作を受けるもの。

オペレーティング・システム：プログラムの実行を制御するソフトウェアの集まり。入出力制御，ファイル管理，記憶域の割り振り，データ編集，デバックの支援などを行うソフトウェアがある。

カーソル：ディスプレイ上の位置を示すために使用される移動マーカー。

改行文字（LF）：印刷または表示位置を次の行に移動させる制御文字。

回線：データ通信において，データの送受信に使用される有線やマイクロ波ビームのような物理的媒体。

階層：ある目的ごとに分けられ，目的に沿ってレベルを持つ構造。

重ね書き：すでに記憶域のある部分にあるデータを壊して，新しいデータを読み込むこと。

仮数：浮動小数点表記された数における小数点以下の部分の数値。

可変長レコード：入力データの長さに応じて長さを変更できるレコード。

環境：DOSが操作をするときに使用する変数に，値を設定しておくことができる32バイト以内の文字ストリングの領域。環境はプログラムの実行にあたって，構成パラメーターを運ぶ機能を果たす。

関数：ある目的に沿った機能を果たすための処理を行い，通常ただ1つの結果を返す処理単位。

記憶域：コンピューターの内部において，プログラムを実行させるためにディスクからプログラムやデータを読み込むことができ，かつプログラムの実行にあたって記憶位置を示して，使用できる領域。

記憶保護：記憶域または，ディスクに対するデータの読み取りや書き込みを禁じること。

共用ファイル：複数の処理プログラムによって同時にオープンされるファイル。

行：文字その他の表現を水平方向に配列したもの。

空ストリング：長さが0で文字が入っていないストリング。

区切り文字：入力行内の語や値をグループ化または分離する文字。コマンドやステートメント中の項目と項目を区切るための文字。

クリッピング：図形処理において指定範囲外の図形を無視し，指定範囲内の図形のみを有効とすること。

クロック：同期をとるために周期的な信号を発生する装置。それぞれの信号は刻時パルスまたは刻時きざみと呼ばれる。

K:1000 を表す。記憶容量に関して使用する場合は，2 の 10 乗（10 進表記で 1024）を示す。たとえば，2K バイトは 2048 バイト。

桁あふれ：演算の結果，所定の記憶域単位の容量を越えた状態。

現行ディレクトリー：ディレクトリーの指定なしでファイル指定をした場合に仮定される省略時ディレクトリー。BASIC を始動したときのディレクトリーが省略時のディレクトリーとなる。

現行ドライブ：ドライブの指定なしにファイル指定した場合に仮定される省略時のドライブ。BASIC を始動したときのドライブが省略時のドライブとなる。

コード：データやプログラムをコンピューターに受け入れられる記号の形式で表現したもの。プログラムまたはその一部。

固定長：長さが変わらないもの。BASIC では，データやレコードの長さに対して用いる。BASIC のランダム・ファイルのレコードは固定長である。

コマンド：特定のプログラムの実行を要求するために外部装置からシステムに対して与える指令。BASIC 言語の構文のコマンド，ステートメントは明確な違いがあまりなく，コマンドであってもプログラムとして記述できるものが多くある。ただし，コマンド本来の機能は，BASIC のプログラムを記述し，保管や実行などの支援を行うことを目的として作られている。

コメント：プログラムの中でプログラムの説明を記述した部分。プログラム

のデバッグや保守に利用する。コメント部分はプログラム実行命令とみなされない。

座標：図形処理を行うにあたり，グラフィック・モードでディスプレイ上の位置を識別する値。

サブルーチン：主となるルーチンの果たす機能の内の一部の機能を有し，主ルーチンの一部として使用することのできるルーチン。

システム・リセット：システムの動作を中断し，DOS を再始動すること。

順次アクセス：ファイルの先頭から順番にレコードを書き込み，検索する場合もファイルの先頭から順番に探していく，入出力の方法。

省略値：何も指定しなかった場合に仮定される値，または自動的に設定される状態。

初期設定：プログラムを記述するときに，変数，配列などで使用する記憶域を最初に設定されているべき任意の値にしておくこと。

書式：フォーマットとも言う。画面やディスレットなどのデータ媒体上における，データの配置またはレイアウト。出力形式である場合もある。

書式送り文字（FF）：印刷や表示の位置を，次の用紙やページなどのあらかじめ定められた位置に移動させる文字。

スクリーン座標系：ディスプレイや画面モードに依存する画面の座標系。物理座標系。

スクロール：旧データが一端から消えると，反対側の端から新データが現れるかまたは新たな入力行が現れるように，表示画像の全体または一部を，垂直方向または水平方向に移動すること。

スタート・ビット：非同期通信において，同期をとるためにそれぞれの 1 文字分のデータの先頭に付加されるビット。1 文字の始まりを示す。

スタック：記憶域中の区域でレジスター情報やサブルーチンのリターン・アドレスを一時的に記憶する場所。

スタック・ポインター：スタックの現在位置を与えるレジスター。

ステートメント：プログラミング言語において，特定の処理を指定するための文章。

ストップ・ビット：非同期通信において，同期をとるためにそれぞれの 1 文字分のデータの最後に付加されるビット。1 文字の終わりを示す。

ストリング：文字列。

制御文字：特定の制御操作（データ伝送の制御，画面やプリンターやファイル出力の制御など）を行う文字。

セグメント：64K バイト（物理セグメント）までの任意の連続したブロック。BASIC のデータ・セグメントとは，最大 64K バイトの BASIC の作業領域を指す。

絶対座標形式：図形処理において，スクリーン座標系の原点（左上隅）を基に点の位置を指定すること。

前景：画面表示における文字や図形。背景以外のもの。

全 2 重通信：データ通信において，双方向に独立して同時に伝送可能な方式。

相対座標形式：図形処理において，ある点からの変位を指定することによって，目的の座標値を指定すること。

装置駆動ルーチン：装置を駆動，制御し，装置との間で直接データの受け渡しを行うプログラム。

増分：BASIC では，プログラム中でループ（繰り返し処理）を行う場合に処理ごとにループ変数に加えられる値。

添字：配列内の要素の位置を識別する番号。

タイリング：模様指定。ビット・パターンを指定することにより模様を描くこと。

端末：情報の送受信ができる装置。一般的には，出力装置としてディスプレイまたはプリンター，入力装置としてキーボードなどを備えた装置。

通信：データの送信または受信。

ディレクトリー：ファイルを管理するのに必要な情報を保持したディスク上の登録簿。

テーブル：データを行や列に配列したもの。2 次元配列。

適用業務プログラム：特定の業務を処理することを目的として記述されたプログラム。アプリケーション・プログラム。

デバッグ：プログラム中の誤りを検出し，追跡し，除去すること。

トレース：プログラムの実行記録。命令が実行された順序を示す。または，プログラムの実行を追跡すること。

DOS：ディスク・オペレーティング・システム。

2 進数：基数 2 の数体系。0 と 1 で数値を表現する方式。

2 の補数：2 進数体系で負の数を表すための方式。

2 バイト・コード文字：連続コード（文字に割り振られている 10 進数の番号）256 ～ 11535 で表現される文字セット。2 バイト・コード文字の連続コード 256 ～ 9091 の文字は，JIS 情報交換用漢字符号系に対応している。表示画面において全角で表示される文字。

ネスト：ある構造をもう 1 つの構造の中に組み入れること。BASIC では，

ループの中に別のループを設定したり，あるサブルーチン内で別のサブルーチンを呼ぶことをネストと言う。

背景：対象を囲んでいる部分。BASIC では，表示画面において，文字や図形以外のまわりの部分。

バイト：コンピューターが情報を処理する単位。本システムの場合，1 バイトは 8 ビットであり，2 バイトは 1 ワード。

配列：1 次元以上の次元で複数の要素（値）を並べたもの。

ハウスクリーニング：ある領域に存在するデータを圧縮し（不要な空間を詰めること），その結果生じた空間を解放すること。

バグ：プログラム内の誤り。

パス：木構造のファイル構成において，特定のディレクトリーまたはファイルを指定するための指定子。一連のディレクトリーとファイル名から構成される。

バックアップ：保存または使用上の対象となるものを安全のために別の媒体に 2 重に保管または複写しておくこと。ディスク内のデータを保全のために別のディスクにも保管しておくこと。

8 進数：基数 8 の数体系。0 ～ 7 までの数字で数値を表現する方式。

バッファー：2 つの異なる装置間のデータ転送において，双方の装置のデータ転送速度や事象発生のタイミングなどが異なる場合，それらを補正するために緩衝機構の役割を果たす領域。

バブル・ソート：並んでいる項目を順番に分類する技法の一つ。項目を 1 対ごとに検査し，順番どおりでなければ交換する。この処理を，並んでいる項目があるだけ繰り返す。

パラメーター：ある目的または処理，特定の業務のために，定数値を与えら

れる変数。BASIC のコマンド/ステートメントに値を与えたり，サブルーチンや関数に値を渡すために使用する変数。

パリティー検査：送信データのテスト方式の 1 つ。通常，全数字の合計が常に偶数（偶数パリティー）または常に奇数（奇数パリティー）になるように 2 進数字のグループに 2 進数字を付け加える。

半 2 重：データ通信において送受信どちらの方向にも伝送可能であるが，交互に 1 方向だけ伝送する方法。

bps：通信における伝送速度を表す単位。

引数：独立変数。たとえば数表内の値を探すとき，希望する場所を示す 1 個以上の数。

ビット：2 進数字。コンピュータで扱う情報の単位。

非同期通信：伝送路に同期の要素がない通信方式。通常，データの各文字に対してスタート・ビットとストップ・ビットを付加して同期をとっているため調歩同期方式とも言う。

ビューポート：スクリーン座標系内の図形表示領域。

ファイル：1 つの処理単位として扱われる関連したレコードの集合。

ファイル指定：本システムでファイルを指定するための記述の決まり。装置名（ドライブ指定子），ファイル名，エクステンションから成る。ファイル名の前にパスを指定することもできる。

ファイル制御ブロック：ファイルを制御（オープン，書き込み，読み出し，クローズ）するために，記憶域上に作成されたブロック。

ファイルの終り（EOF）：ファイルの最終レコードの直後にある終了表示。

フィールド：レコード中の区域。BASIC ではランダム・ファイルにおいて，

レコード内の各項目（フィールド）の配置を決める。

ブール演算：論理演算の 1 つであつて，そのオペランドおよび結果は 2 値のうちの 1 つをとるもの。

フォント：特定の大きさと形をもつ文字集合。またはその字体。

復帰文字：**(CR)** 印刷位置，表示位置を，同じ行の最初の位置に動かす移動させるための制御文字。

プログラム・ファンクション・キー（PF キー）:BASIC では，キーボードの上側にある，PF1 から PF10 までの 10 個のキー。

プロトコル：通信回線における通信データの形式や送受信の手順等の一連の規定。

プロンプト：処理を行うために必要なコマンドやデータの入力を促すために使用者対して送られるメッセージ。

ヘルツ（Hz）：周波数の単位。

編集：原始プログラムまたは文書の内容を作成または修正すること。たとえば行の挿入，削除，変更，再編成など。

変数：与えられた値のいずれをも取り得る記号。

ボー：信号伝送速度の単位で 1 秒当たりの不連続状態の数または信号事象数。

補数：第 1 に指定された数を第 2 に指定された数から引くことによって第 1 に指定された数から導かれる数。たとえば，基数表記法において，第 1 に指定された数は基数のべき乗であるかそれよりも 1 だけ小さい数である。負の数はその数の補数で表現されることが多い。

明滅：画面上の文字の輝度を規則的に変化させること。文字を点滅させるこ

と。

要素：集合の一部。配列の中の各項目。

呼び出し：プログラム，ルーチン，サブルーチンを実行すること。通常，入口条件を指定し，入口点に飛び越すことにより行われる。

予約語：現在使用しているシステムにおいて特別な固定した意味を持つために予約されており，任意にその語が持つ意味を変更できない語。BASICでは，BASICが使用しているコマンドやステートメントの名前を本来の機能以外に使用できない（たとえば，変数名として使用してもBASICはBASIC言語の機能が指定されたと解釈する）。

ランダム・アクセス：データの検索や登録をファイル内のデータ位置を指定することによって行う入出力方式。

リテラル：原始プログラムの中に存在する記号や量で，データを参照するのでなく，それ自信がデータとなっているもの。

ルーチン：一まとまりのプログラム。汎用的にまたは頻繁に使用することができるプログラムの一部。またはプログラムから呼ばれる一連の命令。

ループ：ある条件が真である間，繰り返し実行される一連の命令。

レコード：ファイル入出力を行う場合に1単位として扱われる関連情報の集合。

連結：BASICでは2つのストリングをつなぎ，2つのストリングの長さの和に等しい長さを持つストリングを形成する演算。演算子として"＋"を使用する。

連続コード：BASICで使用する全ての文字に割り振られている10進数の番号。

割り込み：条件付きで自動的に起こす飛び越し。決まった場所に飛ぶととも

に，どこから飛び越してきたか記憶される。

割り込み処理：ある条件で飛び越しが起こった場合に行う処理。

割り振る：特定の作業を実行する場合に使用する資源を割り当てること。

ワールド 座標系：図形処理機能を使用する人が想定している座標系。論理座標系。

目的別索引

あ 行

アクセス禁止
〜を解除する …… 文 292
〜を指定する …… 文 156
時間が経過したときに実行するルーチンを
指定する …… 文 183
色
〜を指定する …… 文 32
〜を塗る …… 文 200
印刷制御文字を指定するには
…… 解 E-10
ウインドーを設定する …… 文 313
エラー
〜・コードとエラーの起きた行番号を
得る …… 文 79
〜発生時処理ルーチンを指定する
…… 文 176
〜発生の模擬やエラー・コードの定義を
行う …… 文 81
円を描く …… 文 22
音継続の処理ルーチンを指定する
…… 文 180
音の発生 …… 文 274
音楽演奏を行う …… 文 209
音楽バッファーの音の数を返す
…… 文 214

か 行

カーソル
〜位置やカーソルの表示形態を指定する
…… 文 153
〜の行位置を求める …… 文 49
〜の桁位置 …… 文 219
拡張エラー・コードを得る …… 文 84
型宣言を行う …… 文 58
かな漢字変換による漢字のキー入力
…… 解 1-6
環境
〜から値を読み取る …… 文 73
〜にパラメーターを設定する
…… 文 71
画面
〜を消去する …… 文 30
〜にデータを表示する …… 文 221
〜にデータを表示する<指定された
書式で> …… 文 224
〜の文字の属性を返す …… 文 264
〜モードを設定する …… 文 267
キー
〜操作に割り当てられているコード
…… 解 D-19
〜割り込みの処理ルーチンを指定する
…… 文 178
〜の定義 …… 文 125
〜の割り込み処理を制御する
…… 文 130
キーボード
〜からデータを読み取る …… 文 111
〜から 1 文字を読み取る …… 文 108
〜からデータを 1 行読み込む
…… 文 144
記憶域
〜を解放する …… 文 26

〜イメージのファイルを読み込む
…… 文7
〜イメージのファイルを保管する
…… 文9
〜から1バイト読み取る …… 文208
〜内のプログラムや変数を消去する
…… 文171
〜に1バイト書き込む …… 文218
〜の未使用空間のバイト数を得る
…… 文96
機械語サブルーチン
〜の開始アドレスを指定する
…… 文60
〜の呼び出し …… 文294
〜の処理手順 …… 解B-1
〜を呼び出す …… 文10
逆正接を求める …… 文3
行番号
〜を自動的に作成する …… 文4
〜を付け換える …… 文246
区点コードによる漢字のキー入力
…… 解1-7
経過時間ごとの割り込み処理を制御する
…… 文290
ディレクトリーを変更する …… 文16
コード体系の詳細 …… 解D-1

さ 行

サブルーチン
〜から戻る …… 文252
〜に行き，戻る …… 文101
〜に行き，戻る＜式の値に従って＞
…… 文177
座標変換を行う＜ワールド座標系とスクリーン座標系間の＞…… 文215
出力行の幅を設定する …… 文310
出力ポートへ1バイト送る …… 文199
処理の繰り返しを設定する …… 文92
式の値を変数に代入する …… 文139
指数を得る …… 文83
自然対数を求める …… 文158
順次ファイル
〜からデータを1行読み込む
…… 文145
〜のデータを変数に読み取る
…… 文114
〜の入出力手順 …… 解A-5
〜への書き込みを行う＜区切り文字付き＞ …… 文318
条件が真のときにループ内の処理を実行する
…… 文308
実行
〜を再開する …… 文44
〜を中断する …… 文280
〜の終了 …… 文70
時刻の読み取りと設定を行う …… 文287
数式をストリングに変換する …… 文281
数値をストリングに変換する …… 文169
数値精度と有効範囲 …… 解2-14
ストリング
〜を繋げるには …… 解2-34
〜から数値に変換する …… 文295
〜型を数値型に変換する …… 文50
〜内の文字の位置を得る …… 文134
〜内の文字を調べる …… 文119
〜の一部を得る …… 文166
〜の置き換え …… 文164
〜のスペース指定 …… 文277
〜のバイト数を得る …… 文138
〜の左から指定したバイト数分の文字を

BASIC インタープリター ユーザーズ・ガイド（上巻）解説編
N:SH18-2120-2

IBM®
日本アイ・ビー・エム株式会社
東京都港区六本木3－2－12 〒106
TEL(03)586-1111(代)

Printed in Japan